k-NN approach for classifying semantic roles via Tai-mapping projections

Written by:

Hector-Hugo Franco-Penya

Supervised by:

Martin Emms

Thesis

Presented to the:

University of Dublin

The Provost, Fellows, Foundation Scholars
and the other members of Board, of
the College of the Holy and Undivided
Trinity of Queen Elizabeth near Dublin

Faculty of Engineering, Mathematics and Science

School of Computer Science and Statistics

in fulfilment of the requirements for the degree of:

Doctor of Philosophy

June 2013

Thesis 10129

Declaration

I, Hector-Hugo Franco-Penya, declare that this thesis has not been submitted as an exercise for another degree at this or any other university and it is entirely my own work except as cited in the references.

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement, and I agree that Trinity College Library may lend or copy this thesis upon request.

Signature :
Name :
Date :

Contents

List of Figures

List of Tables

List of Algorithms

Agraïments
Podziękowania
Agradecimientos
Acknowledgements

Thanks to TCHPC for three hundred thousand hours of computational time for conducting the experiments of this thesis.

The person that I want to thank the most is my advisor Dr. Martin Emms, who not only supervised my work with great diligence and dedication but also demonstrated interest in my own happiness all along my Ph.D.

Dr. Carl Vogel and Dr. Jennifer Foster for a generous and detailed feedback.

Thanks as well to my examiners Dr. Carl Vogel and Dr. Jennifer Foster for a generous and detailed feedback.

A Nine years project: Nine years ago, I decided thatI want to do Phd, and now when I almost I have one I would like to thank everyone who helped me achieve it. I was told that I should not thank people as it can give an impression that they did the work for me, which is not the case. Hence I am rejecting the good advice of not acknowledging the help and advices just to make my work look bigger and will try to honour all those who deserve, and that too in their native language.

First up thanks to Science Foundation Ireland (SFI) for the scholarship, which provided me with funds to complete this PhD, (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College Dublin..

And thanks to every one who directly or indirectly pays taxes to the Irish government because thanks to your taxes my scholarship was funded.

A mi familia: Quiero darle las gracias a mis dos padres por haber promovido activamente todas las estancias de estudio en el extranjero, lo que me ha permitido aprender de muchas culturas. También a mis dos hermanos, as far as I'm aware they are the only two siblings I have, por haber me ofrecido todo su apoyo, toda su atenciń y todos sus recursos econḿicos y propiedades en tiempos difíciles sin pedir o esperar nada a cambio. También a mis tíos y tías, por su apoyo moral. A mi abuelo materno, por haberme inspirado a estudiar ingeniería informática e investigación a través de cincuenta años invertidos en la investigación del movimiento perpetuo. A mi abuela paterna, por haber estado rezando todos los días y prendiendo velas por mi salud y por mis éxitos profesionales y personales.

Thanks to my eternal beloved for her moral support, you are the key of all my success across these years.

To my workmates in Ireland: I want to thank Lilita and Gerard, for being good officemates and often helping me with the queries I had. I'm proud about having worked with both of you, Gerard was able to propose a Ph.D. thesis with a very central research question *"What makes a translated text different from a non-translated one?"* and still had time to play in a musical band. And about Lili who maintained lots of side projects related to CNGL along with the Ph.D.

I also want to thank Stephan, Alfredo, Martin, Francesca, Oscar, Roman, Erwan, Baoli, Jerom, Ilana, Anne and Carl for the participating in the weekly reading group meting. Special thanks to Baoli L., for making the reading group very interesting and intense and encouraging me to choose SRL as topic for my Ph.D. thesis. Special thanks to Jerom J. for good conversations and to Stephan for cheering me up whenever I was feeling down about my Ph.D., good dart matches and proving that if you feel in love with a French woman you can still fly to Paris every second week to maintain a good relationship and completing a really good Ph.D. I think you deserve that Ryanair name one of its planes after you on the Dublin-Paris route ;-). Special thanks to Oscar and Roman for basically their help to solve technical problems, I wish I was as helpful to you as you were to me and I wish you the best from now to the end of your

Ph.D. Special thanks to Alfredo M. for his advice on the Ph.D. and engaging arguments about politics, as it is good to have some one to practice arguing :-).

Thanks to Arun D. for his advices on my thesis, clear discussions about neuroscience and science in general and the odd friendly squash match. Thanks to Derek and Ania for worrying about my happiness during the last days of the Ph.D.

Thanks to Professor Dan Ramenyi for giving a good idea of what constitutes good research, and to Professor Khurshid Ahmad for giving a good idea of how it's applied research during his lectures.

I want to thank Dermot C. for being a good friend and helping me to get motivated again during my depressive phases and during my obsessive ones across the previous three years.

From the CNGL members I wish to thank Cara Green for giving me the job of teaching Spanish at DCU, which I considered crucial for the development of my teaching skills. Thank Tom Irwin for helping me bind the thesis and his help with many small but frequent issues.

Regarding the student societies activities: I want to thank Sean D. and Paul G. for their collaboration in student societies activities, which was certainly of great help of not getting worn down by thinking about PhD alone. Also I wish to thank Professor Ivana Bacik in student societies activities and Professor David McConnell for the same reason plus spending his own time in discussions about moral issues.

Many thanks to Zarima D. and Alena K. for her classes on Russian language.

Thanks to all my students at TCD and DCU, for keep telling me that they like my lectures and making me proud of them by completing their degrees and getting jobs, I can't name all of you but you know who you are.

Regarding all people involved in innovation: I want to thank Dr. John Whelan and Dr. Margaret Woods for giving me a clear picture of the current Intellectual Property policy and to Professor Eoin O'Neill for being a good mentor during the last years.

Regarding the quest of learning useful things along side of the Ph.D.: I want ot thank Fred H, Stephan, Gurpreet Singh, Snega A. and Massimo S. Your help is more appreciated than what words are capable of expressing.

At Poland

To my old workmates in Poland who helped me to keep wishing to do a Ph.D. Many thanks to Professor Boris W. for sharing his passion about science and inviting me to the ship expeditions. Many thanks to Michal J. for good discussions about science over the years, thanks also to Magda, Ola and Kasia. Kasia your passion for high risk sport adventure makes your friendship very interesting.

At UK

To my professors at Hertforshire university for encouraging me to do a Ph.D.: I want to thank Daniel Polani and Chrystopher Nehaniv for their lectures on Symbolic AI and Artificial Life. And thanks to Yi S. for being a patient supervisor and spending her time teaching me how to do things. these are in three different lines.

To my friends for encouraging me to keep studding: Many thanks to Ira for defending higher education and insisting that I should get a Ph.D., for asking scientific questions, and for inspiring technological challenges, your ability to make long toasts in social events, and to find out very quickly what matters to anyone together with your extraordinary professional success makes you a really unique personality. It makes me wonder that if one day computers will be efficient to train people on such social abilities. Thanks again for making me feel or reminding me that I am working on something important and interesting.

To Andrea G., I'm proud of you for deciding to go back to academia to study more and get a better job.

Many thanks to Gurpreet for many things even though I know you don't like to be thanked, but this is my way to honour you. Thanks for being a good collaborative workmate for many long time since we meet at Hertfordshire, for encouraging me to apply to TCD, and finally being a good house mate, in spite of whatever can seems you make me feel more closer to the culture of my home land than any other house mate I had. You are of the very few to feel outrage when your close friends do not eat your food or use your possessions.

Thanks to Kai M. for encouraging me to go ahead with a Ph.D.

At Spain

A mis amigos en España A Barru, c. Alex y c. Pablo por su apoyo y entusiamo en que complete un doctorado. A Alena G. por preocuparse y cuidarme con gran interes para que completara los estudios.

A mis amigos del sindicato de estudiantes Intelec: Gracias a Isidro, Toni y Luis por provocar interest por la universidad como institucion.

A totom qui m'ha ajudat a Valencià: Gracies a Santiago Feliçi, Beatriz Aguirre, Ximo i Joan Pelechano per una ajuda incalculable per a que acabara els studis.

Gracies a Gregorio Martín, per fer creizer el meu interest en estudiar atraves de les seves classes i per demonstrarme que la universitat pot tindre influencia en la societat a traves dels seus articles sobre el trazat the ave i el equip de football.

Gracias al *niño Jesús* (Jesús G.), por ser un buen jefe y por su ayouda moral en aventurarme en ir a Polonia.

Final thanks

To PhD comics, for helping me to laugh about myself and feel better during my Ph.D. time. To the DU dance society, for entertaining me and keeping me in good mood. Thanks to the Hist and Phil societies for organizing good debates and occasionally changing my mind, not many can manage to change my mind with clean and lucid arguments and wonderful experiences.

I also want to thank the disability service for helping me to find a software to check spelling mistakes in my writing with a text-to-speech tool and the counselling service for helping me when I was falling down.

I want to dedicate this thesis to Patricia Ruiz de Azua a good friend whose's academic interests concerned child abuse.

Summary

Semantic role labelling (SRL) is the task of labelling text with a semantic notation in order to identify who did what? When? How? Etc. Once the text is labelled; that information can be used to solve a multitude of other natural language processing tasks.

The purpose of this thesis is to expand the knowledge about how tree alignment algorithms perform in SRL, and expects to help future researchers and engineers to use and combine those algorithms.

The method used to predict the semantic labels consist on aligning the query to a similar sample with semantic annotation, and project the notation. The similar samples are selected by Tai-distance or Tai-similarity, and the alignments are Tai-mappings.

The system was evaluated on seven different languages: Chinese, German, English, Catalan, Spanish, Czech and Japanese. The differences across languages are large regarding size of the data set, domain, set of semantic labels, length of the semantic dependencies, set of syntactic labels, and parser used to generate the annotation. It lead to a large variety of results, showing that Chinese and German seems to be especially suited to these techniques in spite of the expectation from the public evaluation in CoNLL-2009.

The data provide a diverse set of kinds of information which might be used, including four labels for each word, two syntactic labels: POS and dependency relationship to its head word, and two lexical identity labels: Form and Lemma.

The simplest settings ignore lexical identity and just use the geometry, POS and dependency relations. These settings work surprisingly well in contrast with the majority label base line. In the

main approach in which each argument is separately labelled in a tree, the whole tree acts as context. For the simple settings, losing this context and working with the fragment which contains a single argument reduces accuracy.

More complex settings referring to lexical identity and frames out-perform the simple settings. Promoting the use of samples of the same frame increases accuracy, as long as samples from other frames are not banned. Most experimentation was done labelling single arguments in the context of a whole predicate-arguments structure. The sub-tree contains all the arguments of a single predicate, contrasting the results with experiments in which trees contain less context, just one semantic relation per sub-tree, can be observed that with richer node labelling, losing the context of the other semantic relations, and working with smaller sentence fragments increases accuracy.

The use of tree structures leads to a substantially better performance than string structures.

Making k-NN categorisation to take *distance equivalence classes* properly into account works substantially better than selecting samples by distance and randomly separating distance ties.

Distances and similarities are often considered interchangeable. Several different kinds of equivalences are identified and proof provided, and some of them hold and others do not. It turns out that for k-NN and clustering tasks, similarity outcomes may not be reproducible by distance, but any distance outcomes can be reproduced by similarity. Regarding the empirical comparison on SRL between distance and similarity, as similarity deletion and insertion costs was fixed to zero, distance turns out to perform better than similarity.

Chapter 1

Introduction

1.1 Document structure

This document is divided in to nine chapters and two appendices: (1) Introduction (2) Background (3) Underlying methods (4) Tree-SRL (5) Distance outcomes (6) Further parameter refinements (7) Contrasting Distance and Similarity (8) System variants, and (9) Conclusions and future work.

(1) Introduction

This chapter explains the task of Semantic Role Labelling and its applications.

(2) Background

This chapter describes the data set used in this thesis, and other data sets that could have been used it, shows statistics about the data set used of the seven languages in the thesis, exposing the main public Semantic Role Labelling evaluations, and explains the common architecture of a Semantic Role Labelling system as a pipeline of five sub-systems: parsing, filtering, argument identification, argument classification and joint scoring. The Tree-SRL system proposed on this thesis only performs the argument classification step.

(3) Underlying methods

This chapter introduces basic concepts to understand how the system was built: the tree edit distance algorithm and k-NN the machine learning algorithm. And also the metrics used to evaluate the sytem (mainly F1), and justifies the choice of accuracy as an evaluation measure and the McNemmar test for comparing results.

(4) Tree-SRL

This chapter explains the architecture of the proposed SRL system, how a k-NN system is embedded in it, and the different settings that will be used to evaluate the system in future experiments. This chapter is essential for the comprehension of the experiments, results and discussion.

(5) Distance outcomes

This is the first chapter describing results on tree distance. All experiments were tested using the *evaluation* and the *out-of-domain* data set.

(6) Further parameter refinements

This chapter is an attempt to optimise five internal parameters of the system devoting one section to each of them (a) tuning the deletion cost, (b) tuning weights for POS against dependency relations information, (c) tuning weights for lexical against syntactic information, (d) tuning the k of the k-NN system, (e) tuning the cost of using samples from different frames.

All experiments where tested using the *development* data set as the aim is to tune parameters.

(7) Contrasting Distance and Similarity

This is the most theoretical chapter of the whole thesis. It starts by explaining the theoretical differences between Tai-distance and Tai-similarity and ends by giving an empirical comparison of how much those differences affect the Tree-SRL system.

All experiments where tested using the *evaluation* and the *out-of-domain* data set.

(8) System variants

This is the last chapter of experiments. It explains other ways in which the system could have been designed giving some results for such variations.

All experiments where tested using the *evaluation* and the *out-of-domain* data set.

(9) Conclusions and future work

The first section of this chapter is a bullet point compilation of the conclusions of previous four chapters, and the last two sections describe future work and open questions for other researches.

Appendices

This part contains further details of the data set, tables and technical details that were considered redundant in the main parts of the thesis.

1.2 SRL problem

Semantic Role Labeling (SRL) is a natural language processing task. It deals with internal semantic analysis of a sentence as a stand alone entity. However, the task has recently been expanded to inter-sentence level. SRL is the task of identifying and labelling arguments for a certain predicate. The arguments determine events such as 'who' did 'what' to 'whom', 'where', etc in reference to a predicate.

This is a sample of semantic roles:

(1) $[\text{Yesterday}]_{Temp}$, $[\text{John}]_{Cutter}$ **cut** $[\text{the grass}]_{Thing\ cut}$ $[\text{with the scissors}]_{Instrument}$

The same semantic structure can have multiple syntactic representations. The following sentences have an equivalent semantic structure to Sentence 1.

(2) The grass was cut by John yesterday with the scissors

(3) Yesterday, the grass was cut with the scissors by John

(4) With the scissors, John cut the grass yesterday

(5) Yesterday the grass was cut by John with the scissors

(6) John cut the grass with the scissors yesterday

One purpose of SRL is to obtain plain semantic similarity across syntactic alternatives. For instance, a Semantic Role Labelling system would take plain text as input for sentences like Sentences 2 to 6, and produces labelled sentences as output like Sentence 1.

Role-label inventories and annotation principles vary widely, the two main styles correspond to FrameNet and to Propbank. In the FrameNet approach, a concept is identified and a set of roles specified and named for that concept, all of which defines a Frame. Typically several distinct lexical items belong to a single frame. In the PropBank approach, the set of roles is defined for each of the lexical items. There is certain continuity on the set of roles used across the different lexical units, even across lexical units which represent different concepts. Core arguments are enumerated and their meaning is defined in reference to the lexical unit. And adjuncts have a non-enumerated label and their meaning is independent of the lexical item.

Further description of both approaches will appear in next sections.

1.3 Motivation

The motivation for further improvements and developments in the Semantic Role Labelling system is its utility for other Natural Language Processing applications. The promise of SRL is to create smarter natural language processing tools by making computers understand the content of human texts more profoundly.

Producing electricity it is not important *per se*, but it is important because it enables other technologies to flourish such as fridges to keep food fresh much longer time. Computing is important because it enables other technologies to exist like a robust high fidelity global telephone network. In a similar way, Semantic Role Labelling is not just important because it annotates text, but it is important because this annotation enables other natural language technologies to flourish. This section is list of other natural language processing problems that can take advantage of an SRL system.

Information extraction and question answering

Information extraction and question answering are similar tasks which can lever semantic roles to extract answers for the queries. For instance, in the question *"When was Napoleon defeated?"* The pattern to be sought should contain: (this example is from (Went-tau Yih et al., 2006))

- 'defeated' is the predicate.
- 'Napoleon' is the patient of the predicate *defeated*.
- the proposition will have an argument indicating a time in which the action happens and that time is the answer or the field aimed to extract.

This kind of annotation is exactly what a SRL system provides.

For information extraction, the searched pattern would be very similar. (Narayanan and Harabagiu, 2004; Shen and Lapata, 2007) used SRL in question answering systems. (Surdeanu et al., 2003) is an example of an information extraction system which uses a semantic role labelling system.

If a semantic role labelling system is available it is easier to search for a semantic structure with one unknown argument than to search directly in raw text, and the missing argument(s) can be the information being sought.

Information retrieval

Moreda et al. (2005) developed an Information Retrieval system based on SRL. An Information Retrieval system based on string matching would process the query or string *"Harry loves Sally"* by searching for documents containing the words *Harry*, *Sally* and *love*. The problem is that the system would not be able to distinguish among the following sentences (Khoo, 1997):

1. Sally loves Harry, but Harry hates Sally.

2. Harry's best friend loves Sally's best friend.

3. Harry and Sally love pizza.

4. Harry's love for Sally is beyond doubt.

The words 'Harry', 'Sally' and 'love' can occur in the same sentence, but they do not necessarily bear the relationship that the query (*"Harry loves Sally"*) was meant to search for. The previous sample may not be a very realistic case for an information retrieval system. However, it illustrates very nicely that all sentences contain the words in the query, but only one of them contains the meaning of the query.

A semantic role labelling system helps to discriminate among the previous sentences, because the search would be on a semantic structure in which "Harry" is labelled as the one in love and "Sally" is labelled as the object of affection, but not the other way around.

Text summarisation

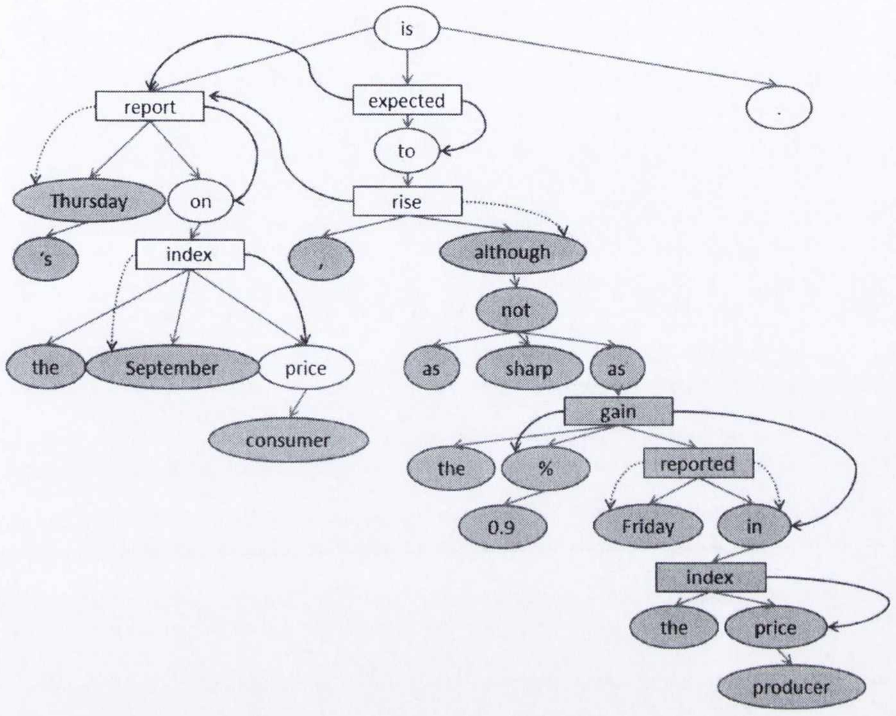
Melli et al. (2005) developed a text summarisation system based on the assumption that predicates and their core arguments have the essence of the sentences. Therefore adjuncts can be removed from a summary. For instance:

(7) Thursday's report on the September consumer price index is expected to rise, although not as sharply as the 0.9% gain reported Friday in the producer price index.

If the following predicates are kept 'raise', 'expected', 'report' and 'index' with their core arguments, the sentence can be summarized into:

(8) The report on the price index is expected to rise.

Figure 1.1 illustrates this sample.



In the proposed summarisation, the white nodes are kept, and the grey nodes are removed. Straight lines represent dependency relations. Square nodes represent predicates. Curved lines represent semantic relations, the solid lines represent core argument and the dotted lines represent adjuncts. (This sentence is the fourth sentence of the trial English dataset used in this thesis).

Figure 1.1: Possible summarisation

Text classification

Shehata et al. (2008) improved a text classification task by basing the classification on the terms of the concepts of a text rather than classifying the terms that appear in the document. For instance, consider the following sentence:

(9) We have **<u>noted</u>** how some electronic techniques, **<u>developed</u>** for the defense effort, have eventually been **<u>used</u>** in commerce and industry.

The SRL system identifies three predicates: the words in bold underlined font. For each of these predicates the rest of the sentence will be labelled as:

(10) [We]$_{ARG0}$ [noted]$_{VERB}$ [how some electronic techniques, developed for the defense effort, have eventually been used in commerce and industry.]$_{ARG1}$

(11) We have noted how [some electronic techniques]$_{ARG1}$,[developed]$_{VERB}$ [for the defense effort]$_{ARGM-PNC}$, have eventually been used in commerce and industry.

(12) We have noted how [some electronic techniques, developed for the defense effort]$_{ARG1}$, have [eventually]$_{ARGM-TMP}$ been [used]$_{VERB}$ [in commerce and industry]$_{ARGM-LOC}$.

Terms like "electronic techniques" appear as arguments for all three predicates. Therefore, they will have more weight than "commerce and industry" which only appears as arguments for the predicates 'noted' and 'used'.

Paraphrasing

Ellsworth and Janin (2007) developed a paraphrasing algorithm in which a SRL system identifies verbs and their arguments which will be re-arranged in the sentence using a different valence for the verb. For instance the sentence:

(13) I want your opinion

can be paraphrased as:

(14) Your opinion is desired

From the Sentences 13 and 14 a SRL system would extract the same semantic structure, but the sentences are different.

Machine translation and its evaluation

The following sample illustrates a translation from English to Farsi, in which the order of the arguments is changed to fit the structure of Farsi language and the arguments are translated one by one.

English	(SVO)	Farsi	(SOV)	
AGENT	The little boy	AGENT	pesar koocholo	boy-little
PRED	kicked	THEME	toop germezi	ball-red
THEME	the red ball	ARGM-MNR	moqtam	hard-adverb
ARGM-MNR	hard	PRED	zaad-e	hit-past

An SRL system can be used to extract the semantic structure of a sentence, then translate the arguments on their own and transform the semantic structure into a text for the target language. For samples of machine translation systems improved by an SRL system see (Boas, 2002; Wu and Fung, 2009).

Giménez and Màrquez (2007) developed a semantic similarity measure between automated and reference translations by comparing their semantic roles, the more similar the semantic structure is between the reference and the automated translation, the better the translation is. A good translation should contain the same semantic structure with the same set of arguments as the original text.

Co-reference resolution

Ponzetto and Strube (2006) developed a co-reference system that use semantic roles as features to learn the task. For instance:

> A state commission of inquiry into the sinking of the Kursk will convene in Moscow on Wednesday, the Interfax news agency **reported**. It **said** that the diving operation will be completed by the end of next week.

In this example, the SRL system will label "the Interfax news agency" as the AGENT of "reported" and "It" will be labelled also as the AGENT of "said". That information helps to predict the a co-reference between both of them.

Textual entailment

Tatu and Moldovan (2005) developed a textual entailment system in which semantic relationships are compared to each other to decide if the meaning of one can be inferred from the other.

As shown previously, Sentence 1 on page 3, can be re-written in many other ways by swapping the order of its arguments. Another example is:

(15) [John]$_{agent}$ **is** [in the garage]$_{Location}$ near the car.

(16) [John]$_{agent}$ **is** near the car [in the garage]$_{Location}$.

Sentences may look very similar but if their semantic arguments are different they are likely to have different meanings,.

(17) [John]$_{agent}$ is in the car near the garage.

Sentences 15 and 16 have the same meaning[1], but 16 and 17, even if they have the same words they differ in meaning. In Sentence 17 John may be getting to the garage by driving, and in Sentence 16 John may have already reached to the garage and got out of the car.

As mention before,instead of plain text the semantic structure is compared then it is easy to identify differences and consequently realize that the meaning of both sentences (Sentence 16 and 17) are different.

Word sense disambiguation

Agirre and Martinez (2001) used semantic labels to disambiguate the senses of some words. For instance in the sentence:

(18) The bad news will **eat** him.

The object of eat fills the experiencer role; this information can be used to constrain the possible senses for the verb *eat*. The most common sense for the word *eat* applies to food taken into the mouth and swallowed, but as news does not have mouth, the system should disambiguate the word "eat"as having a metaphorical sense, indicating of 'him" that his health will be "consumed"in reaction to the bad news. In order to do this, it is necessary a SRL system trying to label "news" as the consumer, so the next module of the word sense disambiguation system can decided if "news" is an agent capable of "eating" or not, and if not, then the system has to search for another possible meaning.

[1]Both sentences are ambiguous, but both can mean the same.

Chapter 2

Background

2.1 Introduction

Semantic Role Labelling as a task and its importance with respect to other application has already been discussed in Chapter 1. This chapter will begin by detailing the corpora/ data set used for thesis and how public data is annotated for SRL, what are the previous public evaluations on semantic role labelling, what is the common on architecture of a statistical SRL system and finally why it why chosen to perform labelling by projecting annotation through alignment.

2.2 Corpora

Throughout the last decade, large corpora have been manually annotated with semantic relationships. The most important annotated corpora available in English are: FrameNet, PropBank and Nombank. The experiments in this thesis were carried out with data in PropBank style.

2.2.1 FrameNet

Fillmore et al. (2001) built the FrameNet Corpus taking sentences from the British National Corpus. It contains more than 10,000 lexical units, more than 825 frames and more than 135,000 sentences. A lexical unit is a pair of word and a frame, Each frame represents a situation involving participants that can perform different roles, each of this roles of the frame has a tag and a description of which role is representing, this is an annotated example for the predicate "retaliation":

(19) [This attack was conducted]$_{Punishment}$ [in]$_{Support}$ ***retaliation*** [for the U.S. bombing raid on Tripoli...]$_{Injury}$

Different words can belong to the same frame: for instance the frame "Commerce" includes the verbs ("sell" and "buy"). FrameNet data was not used on any of the experiments on this thesis, but it is important to mention the corpus because its impact in the literature review.

2.2.2 PropBank

Frame set sample: edge.01 'move slightly'.

label:	role:
Arg0	causer of motion
Arg1	thing in motion
Arg2	distance moved
Arg3	start point
Arg4	end point
Arg5	direction
ArgM	medium

Example:

[QS]$_{Arg0}$ edged [TCD]$_{Arg1}$ [up]$_{Arg5}$ [6 positions]$_{Arg2}$ [to 43rd]$_{Arg4}$ [from 49th]$_{Arg3}$ [in 2009]$_{TMP}$

Figure 2.1: Frame sample

Palmer et al. (2005) built the Propbank by transferring each sentence into propositions adding a semantic layer to the Penn TreeBank and defining a set of semantic roles for each predicate.

Samples in Figures 2.1, 2.2 and 2.3 were taken from (Palmer et al., 2005).

It is difficult to define universal semantic roles for all predicates. Hence, PropBank defines a set of semantic roles for each possible sense of each predicate (frame).

The core arguments are enumerated. Figure 2.1 shows the 'edge.01' frame and a sample sentence where core arguments 'a0', 'a1', 'a2', 'a3', 'a4', 'a5' and

Frame set sample: accept.01: 'take willingly'

label:	role:
Arg0	Acceptor
Arg1	Thing accepted
Arg2	Accepted-from
Arg3	Attribute

Example:

[He]$_{Arg0}$ [would]$_{ArgM-MOD}$[n't]$_{ArgM-NEG}$ accept [anything of value]$_{Arg1}$ [from those he was writing about]$_{Arg2}$. (wsj 0186)

Figure 2.2: Frameset accept.01 "take willingly"

adjunct 'TMP' are present. Although the meaning of the numbered arguments cannot be generalised. Usually for a particular verb, A0 is the Agent, A1 the patient or theme (Palmer et al., 2005).

Verbs can have adjunct-like arguments, this is a list of the label of the adjunct and their meaning[1] examples are taken from (Babko-malaya, 2005).

- AM-LOC: locative this modifiers indicate where some action takes place. It can reference abstract locations. Example:
  "[in his speech]$_{AM-LOC}$ he was **talking** about ...".

- AM-EXT: extent: indicate the amount of change occurring from an action. Example:
  "the price **rise** [by 10%]$_{AM-EXT}$".

- AM-TMP: time: It points when an action took place. Example:
  "we **played** [yesterday]$_{AM-TMP}$".

- AM-DIS: discourse connectives: These are markers which connect a sentence to a preceding sentence. Example:
  .[But]$_{AM-DIS}$ for now, they are **looking** forward to their winter meeting.

- AM-DIR: direction: Directional modifiers show motion along some path. Example:
  Workers **dumped** large burlap sacks of the imported material [into a huge

[1]The prefix "AM" stands of Argument Modifier

Frame set sample: kick.01 'drive or impel with the foot'

label:	role:
Arg0	Kicker
Arg1	Thing kicked
Arg2	Instrument (defaults to foot)

First example

[But]$_{ArgM-DIS}$ [two big New York banks]$_{Arg0}$ seem [*trace*i]$_{Arg0}$ to have kicked [those chances]$_{Arg1}$ [away]$_{ArgM-DIR}$, [for the moment]$_{ArgM-TMP}$, [with the embarrassing failure of Citicorp and Chase Manhattan Corp. to deliver $7.2 billion in bank financing for a leveraged buy-out of United Airlines parent UAL Corp]$_{Arg2}$. (wsj 1619)

Second example:

[Johni]$_{Arg0}$ tried [*trace*i]$_{Arg0}$to kick [the football]$_{Arg1}$, but Mary pulled it away at the last moment.

Figure 2.3: Frameset kick.01 "drive or impel with the foot"

bin]$_{AM-DIR}$, poured in cotton and acetate fibers and mechanically mixed the dry fibers in a process used to make filters .

- AM-REC: Reciprocal: this is the label of reflexives and reciprocals such as himself, itself, themselves, together, each other, jointly, both, which refer back to one of the other arguments. Example
  "But voters decided that if the stadium was such a good idea someone would **build** it [himself]$_{AM-REC}$, and rejected it 59% to 41%."

- AM-PRD: These arguments point that an adjunct of a predicate is in itself capable of carrying some predicate structure. Example:
  "Prior to his term, a teacher **bled** [to death]$_{AM-PRD}$ in the halls, stabbed by a student".

- AM-PNC: Purpose Not Cause, they point the motivation for the action. Example:
  "More than a few CEOs **say** the red-carpet treatment tends them to return to a heartland city [for future meetings]$_{AM-PNC}$".

- AM-CAU: cause: It indicates the reason for an action. Example:

"Pro-forma balance sheets clearly **show** [why]$_{AM-CAU}$ Cray Research favored the spinoff".

- AM-MNR: manner : it specify how an action is performed. Example: "He **works** [well with others]$_{AM-MNR}$" is a manner.
- AM-ADV: general-purpose: These are used for syntactic elements which clearly modify the event structure of the verb in question, but which do not fall under any following cases:
  - Temporally related (modifiers of events) Example: "Treasures are just lying around, **waiting to be picked up**"
  - Intensional (modifiers of propositions): *Probably, possibly*
  - Focus-sensitive: *Only, even*
  - Sentential (evaluative, attitudinal, viewpoint, performatives) like *Fortunately, really, legally, frankly speaking,* or clauses beginning with *given that, despite, except for, if* Example:
    "[Happily]$_{AM-ADV}$, she sang". (can be paraphrased as 'I am happy that she sang')
- AM-MOD: modal verb: They are: *will, may, can, must, shall, might, should, could, would* and "Phrasal modals" such as "*going (to), have (to)*" and "*used (to)*". Example
  "John **does** not [have]$_{AM-MOD}$ to run".
- AM-NEG: negation marker: This tag is used for elements such as "*not, n't, never, no longer*", etc.

There are two other functional tags: EXT (for extent of an argument) and PRD (for secondary prediction).

Some constituents are split, the way to mention that second part for discontinuous arguments is indicated by adding "C-" as a prefix.

for instance:

(20) [By addressing those problems]$_{Arg1}$, [Mr. Maxwell]$_{Arg0}$ said, [the new funds have become "extremely attractive to Japanese and other investors outside the U.S."]$_{C-Arg1}$ (wsj 0029)

In this case Arg0 is "By addressing those problems the new funds have become extremely attractive to Japanese and other investors outside the U.S."

2.2.3 NomBank

NomBank is designed to complement Propbank, it provides argument structure of about 5000 common nouns in the Penn Tree-bank II corpus with a similar annotation to Propbank and both can be used together. For instance a PropBank/NomBank enlightened system could detect that John was hired by IBM from any of the following sentence fragments without further regularization across predicates (Meyers et al., 2004);

(21) IBM appointed John

(22) John was appointed by IBM

(23) IBM's appointment of John

(24) the appointment of John by IBM

(25) John is the current IBM appointee

(26) IBM's appointment of John

NomBank annotates verbal nominalizations (such destruction, knwledge, believer, recipient), adjectival nominalizations (such ability, bitterness) and another 16 classes such relational (father, president), and partitive nouns (set, variety) (Meyers et al., 2004)

Like Propbank the core-arguments are enumerated and their meaning is defined for each frame, with the intention of compatibility. Figure 2.4 shows an example of the predicate "claim" noun form (which could be found in Nombank) and another in a verb form (which could be found in Propbank), showing that the meaning of the arguments and the annotation format across Propbank and Nombank is compatible.

Frame set sample: predicate "claim" sense "assert"

label:	role:
Arg0	Agent
Arg1	Topic

noun base example:

[Her]$_{Arg0}$ ***claim*** [that Fred can fly]$_{Arg1}$

verb base example:

[She]$_{Arg0}$ ***claimed*** [that Fred can fly]$_{Arg1}$

This sample was found in (Meyers et al., 2004).

Figure 2.4: Verb and Noun examples for the predicate "claim" meaning "assert"

2.3 Empirical evaluations

This section is a brief enumeration of previous empirical of shared task evaluations on SRL since 2002. Public evaluations are important because they enable a direct comparison of the performance between the systems of the participants.

1. CoNLL-2004 (Carreras et al., 2004)

   CoNLL 2004 The task consisted of English languague only. The data set was from the Wall Street Journal and its composition was: training set (sections 15-18), development set (section 20) and test set (section 21) and it was annotated in PropBank style. Sentences were given in constituent tree structure representation.

2. Senseval-3 (Litkowski, 2004)
   The task used given sentence, a target word and its frame to identify the frame elements within that sentence and tag them with the appropriate frame element name in FrameNet style.

3. CoNLL-2005 (Carreras and Màrquez, 2005)
   This extended the CoNLL-2004 by evaluating a full SRL parsing. The syntactic trees were given by two alternative parses. The training data was substantially larger than in 2004. An out-of-domain evaluation corpora provided was based on Brown corpus.

4. SemEval-2007 task 06 (Litkowski et al., 2007)
   This was a task based on identifying prepositional phrases not SRL, but since most of the semantic roles are realised in prepositional phases, this task can be seen as the argument identification sub-task of SRL.

5. SemEval-2007 task 09 (Màrquez et al., 2007)
   One of the sub-tasks was SRL, identifying verb arguments and tagging the verb with the semantic-class label. The systems were evaluated in Catalan and Spanish.

6. SemEval-2007 task 17 (Pradhan et al., 2007)
   This shared task included SRL as a sub-task. The data set was annotated using the PropBank set of labels and plus the VerbNet set as well. The evaluation was done in English language.

7. SemEval-2007 task 19 (Baker et al., 2007)
   The data was annotated in FrameNet style. The task also included to identify the frame of the predicates. As an extra challenge the evaluation data set included frames that did not appear in the training data set.

8. CoNLL-2008 (Surdeanu et al., 2008)
   The sentences were given in dependency based representation. The evaluation was done on English.

9. CoNLL-2009 (Hajic et al., 2009)
   The sentences were given in dependency based representation, the annotation followed PropBank style, and the evaluation was done in seven different language. The data set of this public evaluation is the one used for the experiments of this thesis, for this reason next section will explain it in further detail.

10. SemEval 2010 Task 10: Linking Events and their Participants in Discourse (Ruppenhofer et al., 2010)
    SRL task was extended to inter-sentence level, allowing arguments to be found in neighbour sentences. The evaluation was done in English language. The annotation was provided in PropBank as well as FrameNet styles

The reason to select the data set from the CoNLL-2009 shared task evaluation for this theis is that the event took place at the beginning of starting of my PhD, and at that time it was the most recent public evaluation in SRL and also it had the advantage of being carried in multiple languages. Next section will describe it in further detail.

2.4 The CoNLL-2009 Shared Task

This section is a brief review of the Hajic et al. (2009) paper in which the details of the task are explained.

The CoNLL-2009 shared task was dedicated to parsing syntactic and semantic dependencies. The systems presented were tested in seven languages: Catalan, Chinese, Czech, English, German, Japanese and Spanish.

Two tasks were offered:

Joint task: Syntactic dependency parsing and semantic role labelling.

SRL-only task: Syntactic annotation were already provided, so only semantic role labelling has to be done.

And there were two challenges:

Closed challenge: Only the information given by the organisers was allowed to be used.

Open challenge: Any external tool or resource was allowed to be used, except if it contained parts of the test data set.

2.4.1 Data Format

The following graph-sentence samples will follow the notation specified in Figure 2.5

The CoNLL 2009 uses a dependency tree representation: Figure 2.6 shows the differences between constituent tree structure (2.6a) and dependency tree structure (2.6b).

There is a head word for each constituent node. In Figure 2.6a the double lined outbound arrows specify which child node leads to the head word of the

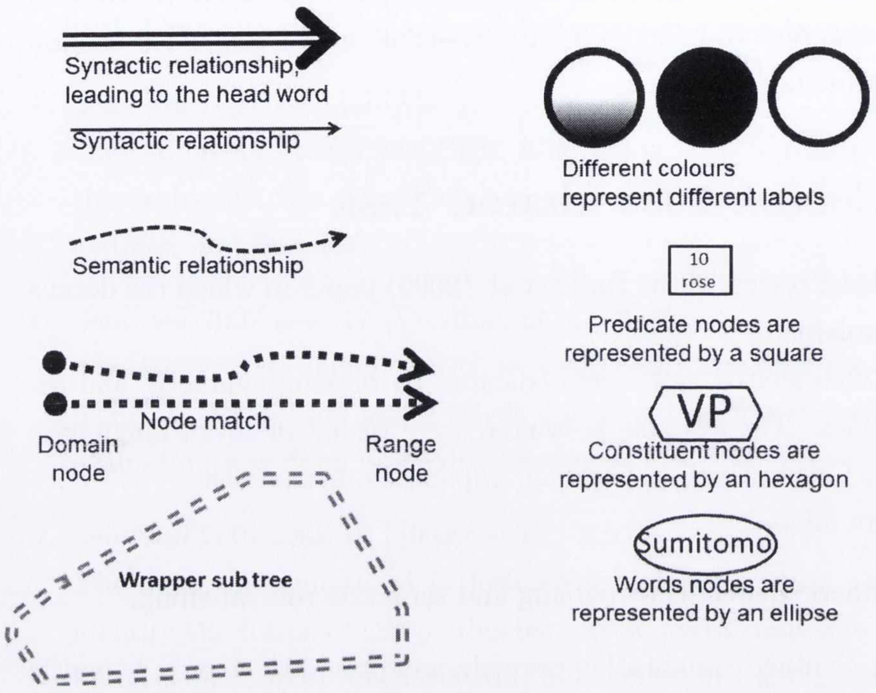

Syntactic relationship, leading to the head word
Syntactic relationship
Semantic relationship
Node match
Domain node
Range node
Wrapper sub tree
Different colours represent different labels
10
rose
Predicate nodes are represented by a square
VP
Constituent nodes are represented by an hexagon
Sumitomo
Words nodes are represented by an ellipse

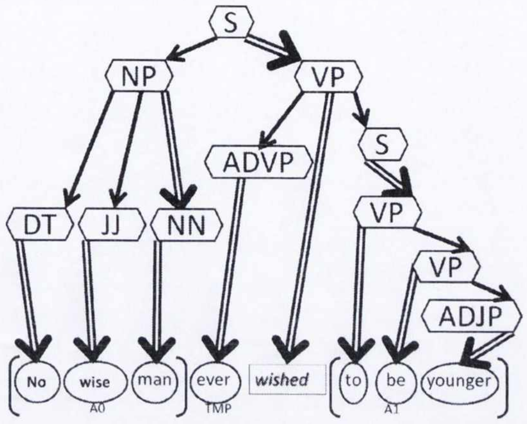
Figure 2.5: Notation for next figures

node itself. Therefore, each constituent node has a head word 'A' and each constituent node has one child node that leads to 'A'. The constituent node can have some children who also have their own head word (nodes 'B_n'). In the dependency tree structure the arrows will go from node 'A' to nodes 'B_n', and the label of the arrow corresponds to the relationship between node 'B_n' and node 'A'. In the sample of Figure 2.6, 'wise' is a noun modifier of 'man'.

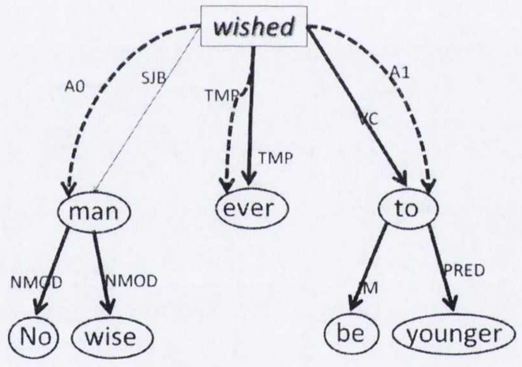
For some of the languages the dependency structures were derived from a constituency structure (e.g. English), for others dependency structures were their native representation (e.g. Czech).

For each word in the data set, the following information is provided: ID, FORM, LEMMA, PLEMMA, POS, PPOS, FEAT, PFEAT, HEAD, PHEAD, DEPREL, PDEPREL, FILLPRED, PRED, and APREDs. (Hajic, 2008):

1. FORM: Word form or punctuation symbol.



(a) Constituent tree structure



(b) Dependency tree structure

The double arrow on Figure 2.6a specifies which child node leads to its head word.
"No wise man ever wished to be younger".
Jonathan Swift
Thoughts on Various Subjects from Miscellanies (1711-1726)

Figure 2.6: Constituent Tree and Dependency Tree structures plus a semantic layer

2. LEMMA: Lemma of word form, or an underscore if not available.

3. POS: Fine-grained part-of-speech tag, where the tag set depends on the language.

4. FEAT: is a set of morphological features (separated by |) defined for each particular language, e.g. more detailed part of speech, number, gender, case, tense, aspect, degree of comparison, etc.

5. HEAD: Head of the current token, which is either an ID value, or zero ('0') if it is the root.

6. DEPREL: Dependency relationship to the HEAD. The set of dependency relationships depends on the particular language. The root node has the label 'ROOT'.

7. PRED: Its value is 'Y' if the word is a predicate, or '_' otherwise.

8. The P-columns (PLEMMA, PPOS, PFEAT, PHEAD and PDEPREL) are the automatically predicted variants of the gold-standard LEMMA, POS, FEAT, HEAD and DEPREL columns. They are produced by independently trained or cross-trained taggers and parsers.

9. APREDs: Columns with argument labels for each semantic predicate following a textual order, i.e., the first column corresponds to the first predicate in PRED, the second column to the second predicate, etc.

2.4.2 CoNLL-2009 data set

This data set is described in further detail because this is the data set used for the experiments of this thesis.

The CoNLL-2009 shared task (Hajic et al., 2009) evaluation was carried in seven languages: Catalan, Chinese, Czech, English, German, Japanese and Spanish. Each language data set was annotated by a different group of annotators without a uniform guide (except Spanish and Catalan) but the organisers of the CoNLL-2009 evaluation adapted the data set format annotation in order to make experiments across different languages comparable with each other. Spanish and Catalan are extraordinarily similar to each other because most of their content is a translation of each other. The data sets are described in great detail in Appendix A because it is the data set used in the experiments of this thesis.

The data set used in CoNLL-2009 Shared task is described in Table 2.1. For each language the data was divided into three or four files: training (T) development (Dev), evaluation (E) and some languages also have an out-of-domain evaluation set (OOD).

The sentences are represented in a dependency tree structure. Each node of the tree is a word, containing four labels which correspond to 'POS' (Part Of Speech), 'DepRel' (Dependency relationship) to its parent node, the 'Lemma' of the word, and the 'Form' which is the word itself. In some cases the dependency tree representation provided, lack of a single root node. This feature makes that graph representation incompatible with the Zhang and Shasha (1989) algorithm for tree edit distance. In order to overcome this issue, an extra artificial node was added to the dependency representation as the unique root of the tree.

Table 2.1 describes how many sentences, predicates and arguments each file of the data set has. It shows the ratio of predicate per sentence and argument per sentence, the size of the files and the branching factor. [2]

The perplexity of a language model p [3] is defined as a function of entropy ($H(p)$) which is a function on the distribution of probabilities:

$$(2.1)\quad Perplexity(p) = 2^{H(p)} = 2^{-\sum_x p(x) log_2(p(x))}$$

where each different value of x corresponds to each word of the language model, and $p(x)$ is the probability of the word x.

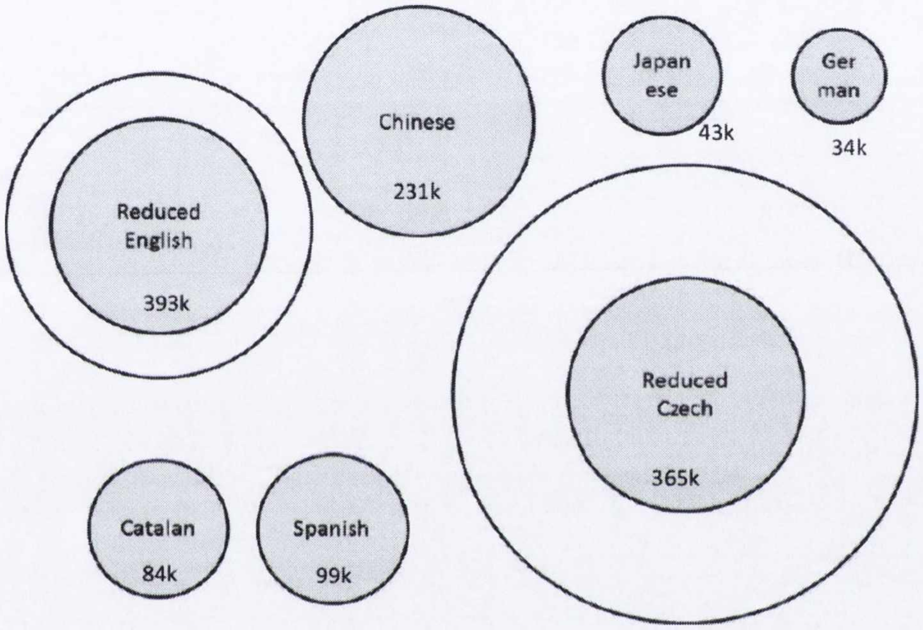
[2]The branching factor ratio corresponds to the sub-trees extracted. Section 4.2 on page 57 describes how sub-trees are extracted, where this measure is defined as the average amount of child nodes of the nodes of a tree excluding leave nodes.

[3]In these case a language model means the probabilities of each word to appear on the text.

		Sentences	Predicates	Arguments	Predicates per Sentence	Arguments per Predicate	file size in KB	branching factor	Average size of sub-tree
Chinese	T	102,809	102,813	231,869	1.00	2.26	40,707	2.182	3.67
	Dev	1,762	8,103	18,554	4.60	2.29	3,318	2.193	3.70
	E	2,556	12,282	27,712	4.81	2.26	4,939	2.172	3.68
German	T	36,020	17,400	34,276	0.48	1.97	42,557	1.835	3.51
	Dev	2,000	588	1,169	0.29	1.99	2,099	1.862	3.48
	E	2,000	550	1,073	0.28	1.95	2,067	1.836	3.47
	OOD	707	648	1,193	0.92	1.84	973	1.784	3.52
English	T	39,279	179,014	393,699	4.55	2.20	60,449	1.646	3.80
	Dev	1,334	6,390	13,865	4.79	2.17	2,125	1.611	3.79
	E	2,399	10,498	23,286	4.38	2.22	3,614	1.644	3.80
	OOD	425	1,259	2,859	2.96	2.27	429	1.728	3.97
Catalan	T	13,200	37,431	84,367	2.83	2.25	43,544	2.254	3.25
	Dev	1,724	5,105	11,529	2.96	2.26	5,937	2.258	3.25
	E	1,862	5,001	11,275	2.69	2.25	5,942	2.255	3.25
Spanish	T	14,329	43,824	99,054	3.05	2.26	4,7406	2.262	3.26
	Dev	1,655	5,076	11,600	3.07	2.29	5,574	2.286	3.29
	E	1,725	5,175	11,824	3.00	2.28	5,611	2.286	3.28
Czech	T	38,727	414,237	365,255	10.70	0.88	87,484	1.444	2.20
	Dev	5,229	55,517	49,071	10.62	0.88	11,753	1.443	2.21
	E	4,213	44,585	39,223	10.58	0.88	9,418	1.452	2.20
	OOD	1,184	16,313	13,882	13.78	0.85	3,441	1.428	2.19
Japanese	T	4,393	25,712	43,957	5.85	1.17	8,931	1.124	5.68
	Dev	250	1,539	2,706	6.16	1.76	539	1.134	6.00
	E	500	3,111	5,316	6.22	1.17	1086	1.126	5.63

In the case of the German data set only a portion of the sentences are annotated, that is why the pericate per sentence ratio is under one.

Table 2.1: Data set statistics



The area of the bubbles represents the size of the dataset by amount of predicates. English and Czech data sets were reduced to the first 20K and 10k sentence in order to be more computational tractable. The small circle represents the reduced version and big circle the original size. Please note that this reduction was not carried for the CoNLL-2009 evaluation but for the experiments on this thesis

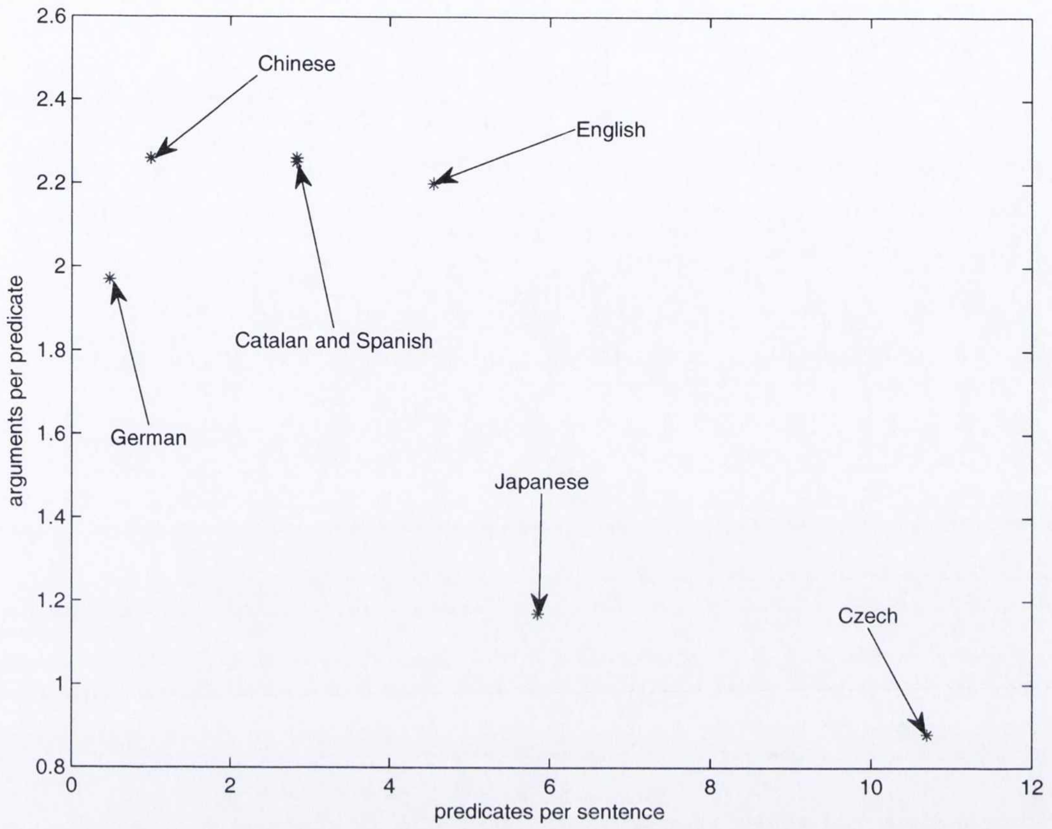
Figure 2.7: Training data set size

Table 2.2: Comparison between the perplexity of the semantic labels and the dependency

The labels presented on the sub-trees correspond to the Training data sets of each language.
No correlation was found between those perplexities and accuracy of the tree semantic role labelling system.

Dataset	Semantic perplexity	Dependency relation perplexity
English	8.02	14.09
Catalan	12.07	8.065
Chinese	7.21	10.93
Czech	11.90	10.19
German	3.38	8.49
Spanish	12.16	8.00
Japanese	9.59	0.05

Table 2.2 shows the perplexity of the amount of semantic labels and the amount of dependency relation labels in the sub-trees of the data sets for the training data set. Perplexity in dependency relations is a measure of how much information the SRL system receives, perplexity in semantic relations is a measure of how much information the system has to produce.



The more annotated a sentence is (number of predicates) the lower the average ratio of arguments per predicate.

Figure 2.8: Arguments per predicate vs predicate per sentence

Figure 2.8 shows the correlation between the ratios of arguments per predicate and predicates per sentence. Czech data has a specially low amount of arguments per predicate[4], this is due to the way the data set was annotated in which almost every word is labelled as a predicate without any arguments.

It is an interesting observation that the ratio of predicates per sentence in Chinese for the training data-set (1.0) is very different to the ratio of the one for the development and evaluation data sets (4.6 and 4.8). It suggests that the partition is not made equally. The same can be said about the German data sets but with smaller differences.

[4]Note that the ratio arguments per predicate is measured in a way in which if a word is an argument of two different predicates it will be counted twice. Therefore the low ratio cannot be the result of control/raising constructions.

2.4.3 Further preprocessing

Apart from reducing the English and Czech data sets to the first 20k and 10k sentences, a new node was added to every singlesentence as a new root node. This decision was taken to solve the problem of having some sentence with multiple root nodes. After this pre-processing step, all old root nodes become the children of a new artificial node.[5] Therefore the sentences had to be converted from a forest structure into a tree structure and make it possible to use tree edit distance over them. No further preprocessing was required.

2.5 Semantic role labelling pipeline

Having reviewed the different public evaluations, now I will look at the common architecture of a semantic role labelling system.

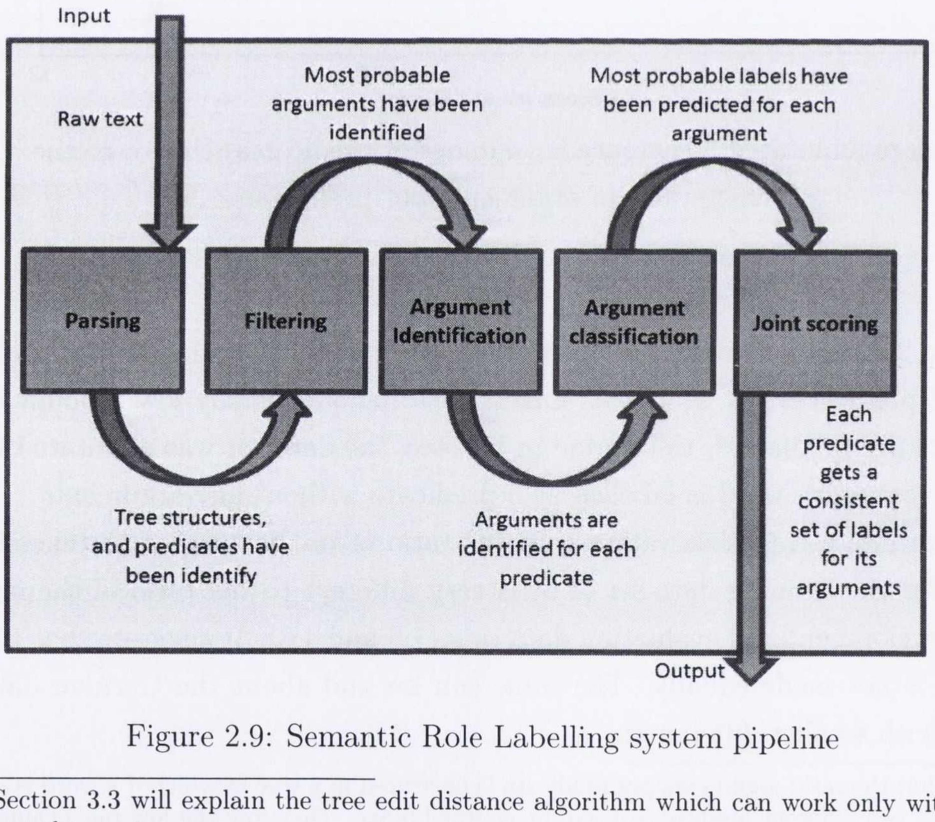

Input
Raw text
Most probable arguments have been identified
Most probable labels have been predicted for each argument
Parsing
Filtering
Argument Identification
Argument classification
Joint scoring
Tree structures, and predicates have been identify
Arguments are identified for each predicate
Each predicate gets a consistent set of labels for its arguments
Output

Figure 2.9: Semantic Role Labelling system pipeline

[5]Section 3.3 will explain the tree edit distance algorithm which can work only with fully connected tree structures

Màrquez et al. (2008) claim that the most common architecture of a Semantic Role Labelling system consists of the following five steps: parsing, filtering, argument identification, argument classification, and joint scoring. Figure 2.9 illustrates these steps, which are explained in the text that follows.

2.5.1 Parsing

Parsing raw text into syntactic representation is an old and well-studied problem. Therefore, most semantic role labelling systems use an external syntactic parser, a black box for pre-processing raw text and some authors do not consider that parsing is part of the semantic role labelling task. According to Palmer et al. (2010, page 46) the most popular parsers are Collins (1999) and Charniak and Johnson (2005).

Carreras and Màrquez (2005) estimated a degradation of 2.18 F1 points by comparing SRL systems using only shallow parsers in contrast of the same systems using full parsers.

2.5.2 Filtering

An argument can be a continuous or discontinuous sequence of words. Therefore any sequence of words in the sentence can be an argument. An exhaustive exploration of all possible arguments is either not feasible or is extremely computationally expensive as the amount of them is very large. Hence, in order to maintain an adequate speed, it is necessary to have a heuristic to prune most of the candidate arguments for the next step (argument identification). This algorithm should be computationally light and should have a very high recall.

Màrquez et al. (2008) claimed that the simple heuristic rules proposed by Xue and Palmer (2004) for constituent tree structures are commonly used to perform filtering because of its success.

Figure 2.10 illustrates the Xue and Palmer (2004) filter algorithm which is shown in detail in Algorithm 1. In Figure 2.10 the predicate of interest is "warned". The system first adds the PP "of tough measures" to the list of candidates[6] as it is a sister node of the predicate. Then moves to the next

[6]Candidate to be an argument node

```
Input: T: constituent tree
Input: S: set of predicates
Output: SC: set of argument candidates per predicate
foreach predicate p in S do
    Pointer = p;
    while Pointer ≠ root(T) do
        foreach Q sibling node of Pointer do
            if Q is coordinated with p then
                continue;
            end
            if Q is a prepositional phase then
                SC(p).add(children(Q));
            end
            SC(p).add(Q);
        end
        Pointer = parent(Pointer);
    end
    return SC ;
end
```

Algorithm 1: Filtering heuristic (Xue and Palmer, 2004), on constituent tree structures

ancestor and adds the NP "Premier Ryzhkov" to the list of candidates to be an argument. At the next level the two 'S's form a coordination structure. Therefore, there are no more constituents to add.

Experimental results show that even using imperfect parsers, the pruning algorithm improves the overall accuracy of the system (semantic role labelling accuracy) (Palmer et al., 2010, pag 33). For Màrquez et al. (2008) an exhaustive exploration will not be feasible because the data is always extremely large and unbalanced.

2.5.3 Argument identification and classification

Argument identification and classification modules are described together because they have a substantial amount of details in common. Both phases can be done in a single step although the results are usually better if the phases are done separately (Màrquez et al., 2008). Machine learning algorithms do not

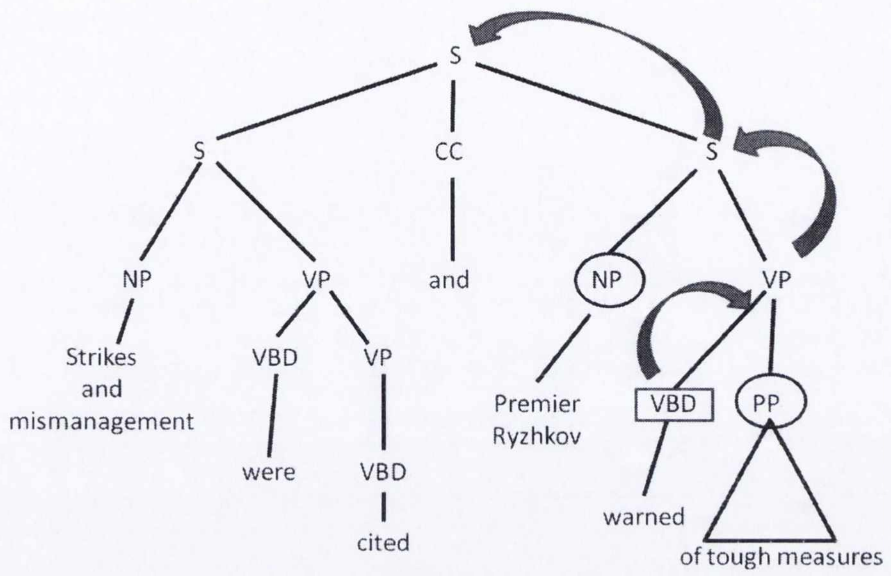


Figure 2.10: Pruning, extracted from (Xue and Palmer, 2004)

handle very unbalanced data well. A palliative solution for that is to split the task argument identification and argument classification. The data for both phases are still very unbalanced but not as much as if it would have processed in a single step.

Balancing techniques (selective reduction and artificial sample creation such SMOTE algorithm (Chawla et al., 2002)) are recommended for these cases but they were not used as they are outside the scope of this thesis.

2.5.3.1 Argument identification

Argument identification is a binary task; a constituent can be a semantic argument or not. Often, sentences have a large amount of constituents, and a very small amount of them are semantic arguments of one of their predicates. In other words, the identification task is extremely unbalanced because there is a small proportion of positive samples (constituents which are semantic arguments for a given predicate) and a large proportion of negative samples (constituents which are not semantic arguments for a given predicate).

Filtering algorithms reduce the majority class which is "this is not an argument", but even using a filtering algorithm, the data are still very unbalanced. As a consequence of machine learning algorithms not handling unbalanced data very well, balancing techniques are recommended. In this step each constituent of a sentence is classified as an argument or not for each predicate although some of them were already classified at the filtering phase.

2.5.3.2 Argument classification

The constituents that have been classified as arguments are passed to a new phase of classification to determine which kind of argument they are. The same machine learning algorithm that was used in the argument identification phase may be used in this phase, but usually using a different set of features; thus, it usually improves the overall performance. This time the problem is not a binary-classification task like in the argument identification phase but multi-classification task, what means that the system has to classify each sample in one of the multiple possible classes (one class for each possible argument).

2.5.3.3 Features for classification

It is common that the features used for identification and classification are the same. Palmer et al. (2010, pages 31-42) describes in detail the features which are most commonly used for classification: Phase type, governing category, parse tree path, position, voice, head word, sub-categorisation, argument set and more recently also: argument order, previous role, part of speech of the head word, named entities in constituents, verb clustering, head word of objects of PPs, first/last word/POS in constituent, constituent order, constituent tree distance, constituent context features and temporal cue words.

Also when dependency tree structures are available other features are used as well(McDonald et al., 2006) , such: Do any of the dependent of the siblings share its POS with the Edge? POS tag of each intervening word between head and dependent. Do any of the words between the head and the independent have a parent other than the head? Are any of the words between the head and the dependent not a descendant of the head? How many children do the dependent have? What are morphological features do the grandparent of the dependent? And have the dependent and its grandparent identical values?

2.5.4 Joint prediction

Several authors designed their SRL systems to first estimate the probabilities of possible labels for each identified argument as if the labels were independent of each other. Then as a second phase, the SRL system would re-estimate the probabilities of all arguments of a single predicate to produce the best overall prediction knowing that the labels are dependent on each other.

This module also helps the system to improve the consistency of the output. For instance it can guarantee the non doubling of core arguments for a single predicate: If an argument is discontinuous the second part would get a different label than the first part, for instance C-A0 would mean the continuation of the argument A0, but the argument A0 can't be assigned to two different arguments in a single predicate.

Gildea and Jurafsky (2002) made the first system based on re-ranking the probabilities of the arguments for each all arguments of a single predicate at the same time. The probabilities for each individual constituent are combined with a probability for the set of roles that will be predicted for each predicate. An interesting example was produced by Toutanova et al. (2005), using maximum entropy. The features used in the re-ranking module based on maximum entropy model included all the features from the individual argument classification module, the sequence of labels for the entire sentence (excluding adjuncts and including the verb itself), and the sequence of unpredicted labels, which simply counts the number of arguments to the left and right of the verb(Palmer et al., 2010, pages 44-45).

2.6 Research motivation

2.6.1 Efficiency

The semantic role labelling system presented in this thesis is based on k-NN which is a lazy learning algorithm. This decision is not shared with most other SRL systems, which opted for fast response machine learning algorithms such as SVM. As a consequence, the system is too slow for a practical application. It does not mean that the algorithms investigated in this research are useless; the methods presented in this work are used in tasks where the computational time is not critical. For instance, Furstenau and Lapata (2011) used graph algorithms to project labels from one sentence to another, with an even more computationally intensive algorithm than the Zhang and Shasha (1989) one. This algorithm and its efficiency will be described in detail in Section 3.3.2. Furstenau and Lapata (2011) shown that "labelling by alignment" algorithms are adequate for labelling a corpus which would be used as an expansion for the training data set of another SRL system, which will be more time efficient.

Corpus expansion algorithms like the one developed by Furstenau and Lapata (2011) are usually evaluated by the increment on the accuracy of the second system which is using that extended corpus. It leaves an open question: *What is the accuracy of the corpus expansion system itself and what would happen if that system were forced to label all sentences not just choosing the ones for which the confidence is higher?*

2.6.2 Alignment methods

There are a several pattern recognition scenarios that have the characteristics that one has structured test data such a sequence, a tree, a graph or a grid within which some annotation is missing, and one wants to infer the missing annotation by exploiting fully annotated training data.

A possible approach is to seek to define alignments between test data with missing annotation and training cases fully annotated, and to use these to project annotation from the training to test cases.

This has been successfully used in computational biology, for example, to project annotation via sequence alignments (Marchler-Bauer et al., 2002) and graph alignments Kolar et al. (2008).

Semantic Role Labelling (SRL) can be seen as a another instance of this pattern recognition scenario which is the target of this research.

2.7 Conclusion

Any Semantic Role Labelling system based on Machine Learning needs a data set to learn from. Therefore, this chapter described the main public corpora resources: FrameNet, Propbank and Nombank. It continues by mentioning the public evaluations of SRL systems since 2002, giving more details on the CoNLL-2009 SRL task, because the data set of this evaluation is the one used for the experiments conducted during the research and presented in thesis.

Then the attention goes to the general architecture of a SRL system, observed on those public evaluations. Those systems are commonly built as a Pipeline of the following five sub-systems: parsing, filtering, argument identification, argument classification and join scoring. The argument classification subsystem is commonly implemented by extracting vectors of features and using a machine learning algorithm to classify. In contrast this thesis explores another way to build the argument classification sub-system, based on extracting subtrees and projecting the annotation from one sentence to another, which will be explained in the Chapter 4 after explaining the evaluation metrics, the tree edit distance algorithm that will be used to project the annotation from the training data set to the evaluation data set, and Machine learning algorithm used: k-Nearest Neighbours.

Chapter 3

Underlying methods

3.1 Introduction

This chapter introduces the evaluation metrics in which using the system was evaluated. It also introduces the two algorithms required to understand the proposed SRL system. These two algorithms are: Tree Edit Distance algorithm, used to align tree sample in order to project annotations and also to calculate a distance score, and k-NN algorithm, a machine learning algorithm used with SRL system which uses the distance scores. How these two algorithms complement each other will also be explained later in the chapter.

3.2 Evaluation metrics

This section explains why accuracy measure was choose as evaluation metric when the CoNLL2009 adopted F1 measure, and introduces the McNemar test which is suitable to compare different versions of the same system classifying the same data set.

3.2.1 Measures

In CoNLL 2009, F1 measure was the metric used to compare different systems performances. The Tree-SRL system only performs the argument classification and not argument identification. Therefore accuracy was used to measure its performance (as the values will be the same as F1).

(3.1) $$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

(3.2) $$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

(3.3) $$F_\beta = \frac{(1+\beta^2) * Precision * Recall}{\beta^2 * Precision + Recall}$$

Where:

- A True Positive is a correct prediction.
- A False Positive is produced when the system predicts a wrong label or a label in a wrong place.
- A False Negative is produced when the system misses a label or when it predicts a wrong label.
- β is an arbitrary number to weight precision and recall, usually it is set to 1.

Imagine that the following sentence is properly annotated:

(27) Yesterday [Microsoft shares]$_{object}$ raise [20%]$_{amount}$

and the following three predictions of an SRL system:

(28) Yesterday Microsoft shares rose [by 20%]$_{amount}$

(29) [Yesterday]$_{original.price}$ [Microsoft shares]$_{object}$ rose [by 20%]$_{amount}$

(30) Yesterday [Microsoft shares]$_{object}$ rose [by 20%]$_{original.price}$

Sentence 28 contains a False Negative, because "Microsoft shares"where not labelled. Sentence 29 contains a False Positive, because "Yesterday" was labelled as "original.price" it should not be annotated[1], and Sentence 30 contains two errors: a False Positive, because "20%" was labelled as "original.price" and a False Negative, because "20%" should be labelled as "amount".

In the case of only labelling arguments and the system being forced to make a prediction, a wrong prediction (label an X sample as Y) is counted twice (like Sentence 30), one as a false positive (an error Y) and one as a false negative (a missed X). Therefore, the amount of true positives is the same as the number of correct predictions, and false positives and negatives are equal to each other and have the same amount as wrong predictions. Expressed in a mathematical notation:

$$TP(True.Positive) = RL(Right.Labels)$$

$$FP(False.Positive) = FN(False.Negative) = WL(Wrong.Labels)$$

$$A(Accuracy) = \frac{RL}{RL + WL}$$

$$Precision = Recall = \frac{RL}{RL + WL} = A$$

$$F_\beta = \frac{(1+\beta^2) * A * A}{\beta^2 * A + A}$$

$$F_\beta = \frac{(1+\beta^2) * A * A}{(1+\beta^2) * A} = \frac{A * A}{A} = Accuracy \tag{3.4}$$

The conclusion is that in this particular case the values of F_β measure and accuracy are the same, so accuracy is adopted as the main parameter to compare the performance of the systems. It is true that the SRL system can produce an invalid label. Such cases should be considered False Negatives but not a False Positives. However, those cases are so rare that. Hence it is assumed they never happen, and consequently they were count as wrong predictions as well.

[1]This cases assumes that "yesterday" is not an argument of "raise".

3.2.2 McNemar test

The problem of labelling the semantic relationship between two identified constituents is a classification one. If the system is forced to make a prediction, the McNemar (1947) test is adequate to determine if a modification in the system settings produces a statistically significant change on the output of the system.

The test can be explained as if it were applied to a contingency table like the one shown in Table 3.1:

Table 3.1: McNemar contingency table

	Correct by classifier 2	Incorrect by classifier 2
Correct by classifier 1	a	b
Incorrect by classifier 1	c	d

The null hypothesis is that the probability of c is the same as the probability of b.

The values for the McNemar (1947) test statistic with Yates (1934) correction for continuity is given by Equation 3.5:

$$\text{(3.5)}\quad \chi^2 = \frac{(|b-c|-0.5)^2}{b+c}$$

χ^2 can be approximated by a chi-squared distribution with 1 degree of freedom. If $b + c < 25$, the χ^2 distribution is not a good approximation. In such cases no statistical difference was claimed. If χ^2 is significant, a statistical improvement is claimed.

In this thesis, the McNemmar test was considered significant for a p value of chi-square of 0.05 and 0.001 for strictly statistically significant. Furstenau and Lapata (2011) used the same test to detect statistical differences with the same p values.

3.3 Tree edit distance

This section explains the tree edit distance algorithm which is a central piece of the architecture of the proposed SRL system of this thesis. It produces an alignment that can be used to project annotation and it also produces a distance score which can be used to train a machine learning algorithm. It will start introducing Levenshtein distance, which is the string version of tree edit distance, and Tai mappings which contain the restrictions of the alignments produce by the algorithm.

3.3.1 Levenshtein distance

The Levenshtein (1966) distance is a string metric for measuring the divergence of two sequences and map one into the other. It is defined as the minimum number of edits needed to transform one string into the other, by counting operations for insertion, deletion and substitution of a single character. This algorithm is a predecessor of tree edit distance.

The algorithm constructs a bi-dimensional matrix with the size of the first string plus one by the size of the second string plus one. Each cell of the matrix corresponds to the cost of deletion/insertion of all items of the sub-string from the beginning of the stings to its own position. The first row and column is an ascending sequence 1, 2, 3 .. n, and the other cells are defined as the minimal cost between the following options:

- a deletion: d[i,j]= d[i-1,j]+1;
- an insertion: d[i,j] = d[i,j-1]+1;
- a substitution : d[i,j] = d[i-1,j-1]+1 ; only if s_1[i] != s_2[j]
- a perfect match: d[i,j] = d[i-1,j-1] ; only if s_1[i] == s_2[j]

Note that the last two options are a single option together, because a perfect match is a substitution where both nodes are equal. The only difference is the cost which is 0 for perfect matches and one for any other operation.

Figure 3.2 illustrates the process of building a matrix by an example.

Table 3.2: Example of levenshtein distance matrix, for the words '*synthetic*' and '*syntactic*'

The words 'synthetic' and 'syntactic' differ in two units in Levenshtein distance, the result is found at the bottom right part of the table. Each deletion, insertion or imperfect swap counts as one unit in Levenshtein distance.

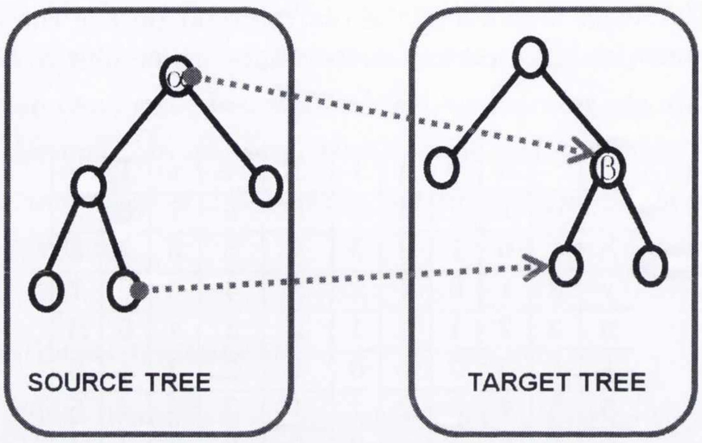
		S	y	n	t	**h**	**e**	t	i	c
	0	1	2	3	4	5	6	7	8	9
S	1	0	1	2	3	4	5	6	7	8
y	2	1	0	1	2	3	4	5	6	7
n	3	2	1	0	1	2	3	4	5	6
t	4	3	2	1	0	1	2	3	4	5
a	5	4	3	2	1	**1**	2	3	4	5
c	6	5	4	3	2	2	**2**	3	4	4
t	7	6	5	4	3	3	3	2	3	4
i	8	7	6	5	4	4	4	3	2	3
c	9	8	7	6	5	5	5	4	3	2

3.3.2 Tree distance

Tai (1979) introduced a criterion for matching nodes between tree representations, and Zhang and Shasha (1989); Shasha and Zhang (1990) developed an algorithm that finds an optimal matching tree solution for a given pair of trees. The importance of the algorithm is that its computational cost is $O(|t_1| * |t_2| * min(depth(t_1), leaves(t_1)) * min(depth(t_2), leaves(t_2)))$ and its spatial cost is $O(|t_1| * |t_2|)$.

It has been applied with some success in several fields of Natural Language Processing such as:

- Text Entailment (Kouylekov and Magnini, 2005),(Negri et al., 2008), (Mehdad, 2009),
- Question Answering (Punyakanok et al., 2004a,b), (Emms, 2006b),
- Parser Evaluation (Emms, 2005b, 2008),
- Question Clustering (Emms, 2006a),
- Question Categorization Emms (2005a).



A map "f" defines a correspondence between the set of nodes of a source tree and the set of nodes of a target tree. The image of the α node is β: $f(\alpha) = \beta$

Figure 3.1: Map example

3.3.2.1 Tai mappings

A Tai map is a mapping between the nodes of two trees. It can be define as:

- S is the set of nodes of the source tree.
- T is the set of nodes of the target tree.
- $f()$ is a map, $f : S \rightarrow T$ where S is the domain and T is the range .

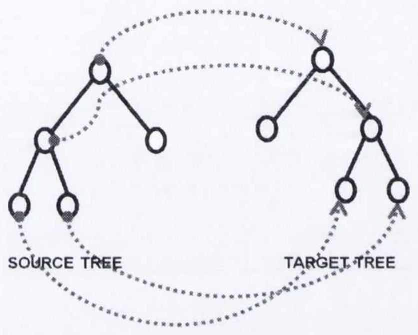
If $(f(a) = x) \wedge (f(b) = y)$, the Tai mapping restrictions can be defined as:

1. **One-to-one:** One node cannot be matched with more than one node.

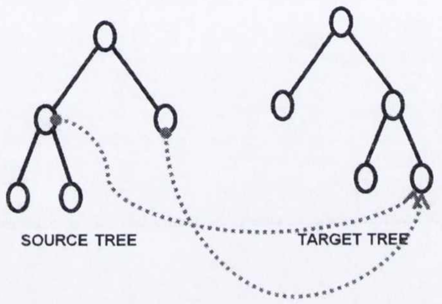
   $a = b$ iff $x = y$.

2. **Left-to-right order preserved:** If a node "a" is on the left of "b", they cannot swap after the mapping.

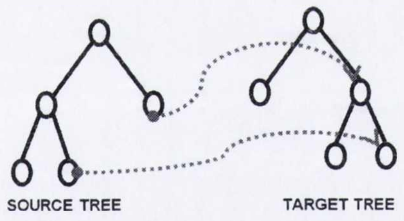
   $a < b$ iff $x < y$.



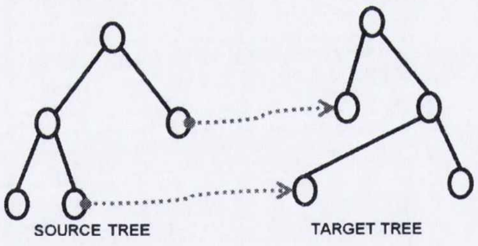
(a) Tai map



(b) Multiple node match



(c) Ancestry not preserved



(d) Left to right order not preserved

Figure 3.2: Tai-mapping restrictions

3. **Ancestor order preserved.** $ancestor(a, b)$ iff $ancestor(x, y)$.[2]

Figure 3.2 shows a sample of Tai mapping(3.2a), and samples of impermissible Tai mappings where one-to-one node is not preserved (3.2b), ancestry is not preserved (3.2c), and left to right order is not preserved (3.2d).

In order for assigning a score to a Tai (1979) mapping it is convenient to identify three sets:

$\mathcal{M}$	the $(i, j) \in \alpha$: the 'matches' and 'swaps'
$\mathcal{D}$	the $i \in S$ s.t. $\forall j \in T, (i, j) \notin \alpha$: the 'deletions'
$\mathcal{I}$	the $j \in T$ s.t. $\forall i \in S, (i, j) \notin \alpha$: the 'insertions'

$\mathcal{M}$ is just the mapping of the S nodes into the nodes of T. $\mathcal{D}$ and $\mathcal{I}$ are just the remaining nodes of S and T which are not 'touched' by the mapping. Let $(.)^\gamma$ give the label of a node and let C^Δ be a 'cost' table, indexed by $\{\lambda\} \cup \Sigma$, where Σ is the alphabet of labels, which assigns 'costs' to $\mathcal{M}$, $\mathcal{D}$ and $\mathcal{I}$ according to[3]:

for $(i, j) \in \mathcal{M}$	cost is $C^\Delta(i^\gamma, j^\gamma)$
for $i \in \mathcal{D}$	cost is $C^\Delta(i^\gamma, \lambda)$
for $j \in \mathcal{I}$	cost is $C^\Delta(\lambda, j^\gamma)$

Where $\alpha : S \mapsto T$ is any 1-to-1 mapping from S into T, define $\Delta(\alpha : S \mapsto T)$ by:

Definition 3.1: ***Distance scoring of an alignment***

'distance' scoring of an alignment

$$\Delta(\alpha : S \mapsto T) = \sum_{(i,j)\in\mathcal{M}} C^\Delta(i^\gamma, j^\gamma) + \sum_{i\in\mathcal{D}} C^\Delta(i^\gamma, \lambda) + \sum_{j\in\mathcal{I}} C^\Delta(\lambda, j^\gamma)$$

From this costs of alignments, a 'distance' score on tree pairs is defined by minimization:

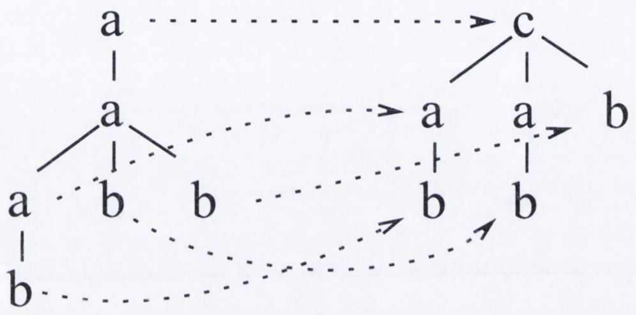
[2] $ancestor(a, b)$ should be read as: "a" is an ancestor of "b".

[3] Note that in this general setting even a pairing of two nodes with identical labels can make a non-zero cost contribution

Definition 3.2: ***Distance scoring of a tree pair S and T***

The Tree- or Tai-distance $\Delta(S,T)$ between two trees S and T is the minimum *value of $\Delta(\alpha : S \mapsto T)$ over possible* Tai-*mappings from S to T, relative to a chosen cost table C^{Δ}.*

There is an illustration of the definitions in Figure 3.3.



With
$C^{\Delta}(x,\lambda) = C^{\Delta}(\lambda,x) = 1,$
$C^{\Delta}(x,x) = 0,\ C^{\Delta}(x,y) = 1$ *for* $x \neq y$,
the alignment has score $\Delta(\alpha) = 3$
and this is minimal for the given C^{Δ}

Figure 3.3: An illustration of tree distance.

$\Delta(S,T)$ can be computed by the algorithm of Zhang and Shasha (1989). Sequences can be encoded as vertical trees, and on this domain of trees the tree distance coincides with a well known comparison measure on sequences, the (alphabet-weighted) string edit distance (Wagner and Fischer, 1974; Gusfield, 1997).

I had formulated the definition[4] in terms of costs applied to mappings which respect tree-ordering properties. In contrast to this declarative perspective, there is a procedural definition via the notion of an *edit-script* of atomic operations transforming S to T in a succession of stages. For both sequences and trees the mapping-based and script-based notions coincide (Wagner and Fis-

[4]The literature contains quite a number of inequivalent notions, all referred to as 'tree distance'; in this Chapter Definition 3.2 will be understood to define the term.

cher, 1974; Tai, 1979; Kuboyama, 2007), and so I omitted further details of the definition via edit-scripts.

While the correctness of the Tai 'distance' Zhang and Shasha (1989) algorithm[5] does not require the cost-table C^{Δ} to satisfy any particular properties, some settings of C^{Δ} clearly make little sense. The combination of deletion/insertion cost-entries which are *negative* ($C^{\Delta}(x, \lambda) < 0$, $C^{\Delta}(\lambda, y) < 0$) with swap/match cost entries which are *not negative* gives the counterintuitive effect that a supertree of S is 'closer' (in the sense of having a lower Δ score) to S than S itself[6]. This is a rationale for the following non-negativity assumption:

(3.6) $\forall x, y \in \Sigma(C^{\Delta}(x, y) \geq 0, C^{\Delta}(x, \lambda) \geq 0, C^{\Delta}(\lambda, y) \geq 0)$

which is a pretty universal assumption, and from which it follows that $\Delta(S, T) \geq 0$, giving a minimum consistency with the everyday notion of 'distance'. In what follows I will confine attention to 'distance' Δ based on a table C^{Δ} which satisfies at least (3.6).

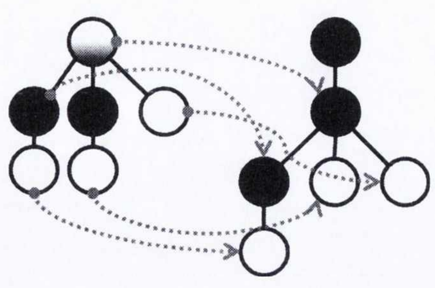
When the cost-table $C^{\Delta}(x, y)$ is constrained more strictly than this to satisfy all the conditions of a *distance-metric*, then it is well known that $\Delta(S, T)$ will also be a distance-metric. Whether such further restriction is desirable is moot: in so-called stochastic variants (Ristad and Yianilos, 1998; Bernard et al., 2008; ?), in which the entries in C^{Δ} are interpreted as negated logs of probabilities,[7] these additional distance-metric assumptions are not fulfilled. The present studies only assume the cost-table C^{Δ} satisfies the non-negativity requirement of distance.

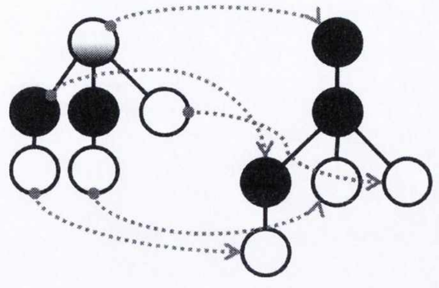
Finally, as an important note it should be mentioned that for a pair of trees there could be more than one mapping at the same distance, see sample in Figure 3.4. The implementation used in the experiments of this thesis promotes the swaps for equal cost over deletions/insertions. Therefore, as the nodes are processed in 'traversal post order', mapping (a) will be prevalent over mapping (b). This feature will have an important impact in the results of the system when the cost of deleting and inserting two nodes would be the same cost as

[5]ie. that it truly finds the *minimal* value of $\Delta(\alpha : S \mapsto T)$ given cost-table C^{Δ}

[6]or a subtree

[7]Therefore, the values are positive.


(a) Alignment 1


(b) Alignment 2

For this sample there are two different Tai maps for the minimal three cost.The different colours of the nodes mean different labels.

Figure 3.4: Isert alignments sample extracted from (Isert, 1999).

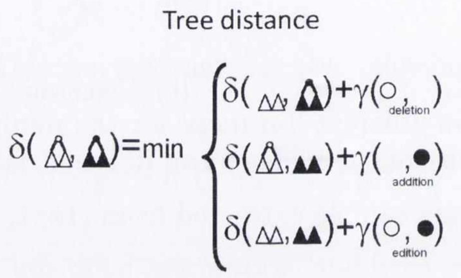
swapping both of them.

3.3.2.2 Zhang and Shasha algorithm

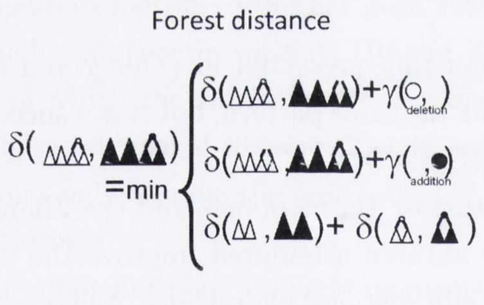
Figure 3.5 shows a recursive description of how to find the cost of minimal Tai map. Some modifications allow not just the cost of the map to be found, but also the map itself.

The Zhang and Shasha algorithm presented in (Zhang and Shasha, 1989; Shasha and Zhang, 1990) follows the same pattern, but it is a successful attempt to make it dynamic (not recursive).

Since the Tai (1979) definition of Tai mappings and the Zhang and Shasha (1989) implementation, a few authors attempted improve the computational efficiency by creating new tree edit distance algorithms. As far as the author of this thesis is aware, there are three algorithms that are more efficient than the Zhang and Shasha (1989) one which also produce the same mappings such as: (Klein, 1998; Demaine et al., 2009; Pawlik, 2011). Those algorithms are more complex than the Zhang and Shasha (1989) one therefore to keep the system simple they were not used. Speed testing is beyond the scope of these thesis.



(a) Tree Distance



(b) Forest Distance

The function $\delta(A, B)$ gives the forest/tree cost/distance between A and B.
The function $\gamma(x, y)$ gives the atomic cost of modifying a node x into a node y.

Figure 3.5: Tree distance

3.4 k-NN algorithm

3.4.1 Introduction

k-NN: k-Nearest Neighbour is the machine learning algorithm of the proposed SRL system of this thesis, this algorithm is a *lazy-learning* algorithm because all computation is deferred until classification, its high computational cost may not be suitable systems in which the time response is critical but it is very adequate for the proposed SRL system because it can select a set of training samples for each query. The proposed system works by aligning samples and projecting labels, that is why it is important to use a machine learning algorithm that selects a set of samples, this would not be possible with the popular machine learning algorithms such Support Vector Machine or Maximum Entropy. In the other hand, k-NN is substantially more computationally expensive than those methods (SVM and ME)[8].

Tree edit distance algorithm produces an alignment and a distance value. The distance value, can be used to generate a ranking of neighbours, and the alignment can be use to make those neighbours to project a label as different parts of the sample will contain different labels, some projections may not produce any label and they will be counted as null votes, this process will be detailed in Chapter 4.

The k-nearest neighbour algorithm works by classifying a sample by a majority vote of the labels of its neighbours. Figure 3.6 illustrates how the mechanism works.

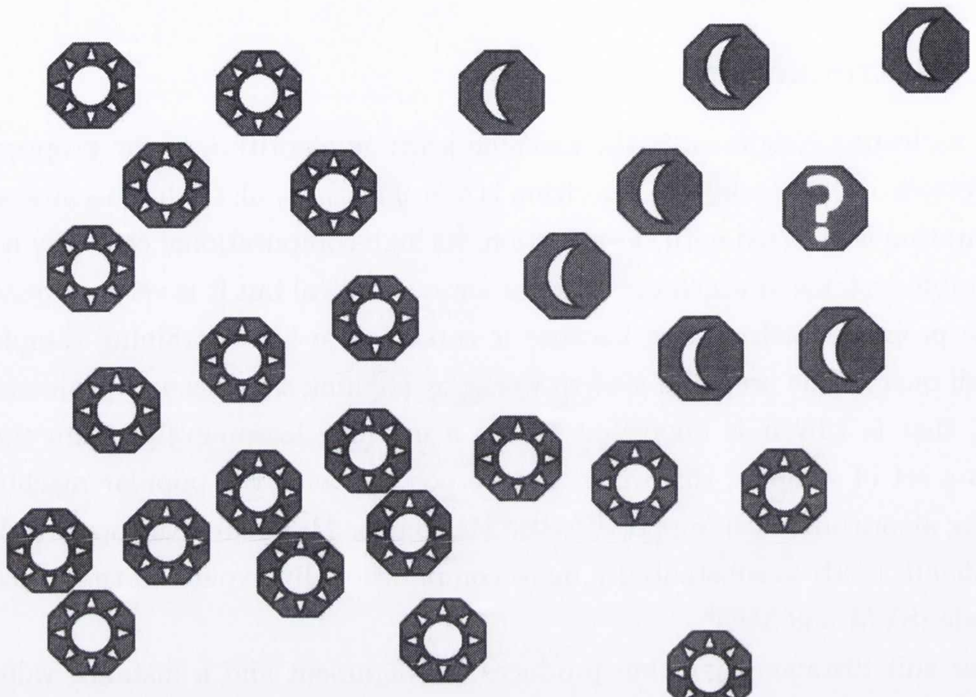
In the implementation used the parameter k means the amount of neighbour samples that are recommended to be used in the prediction.

There are several reasons why a particular fixed value of k may be inappropriate:

1. There are several neighbours at the same distance equally eligible to vote, figure 3.7 illustrates this phenomenon.

2. There is a draw on the voting, figure 3.8 illustrates this phenomenon.

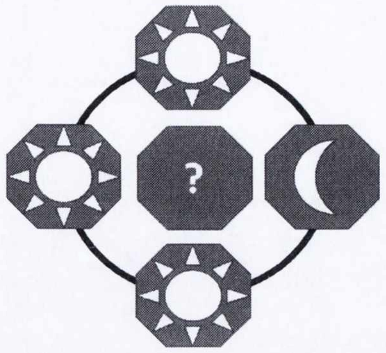
[8]See Appendix B.4 for more details about the computational cost of the tree-SRL system, which is based on k-NN.



The item to be classified (the one with the question mark) is assumed to be more likely to be a moon than a sun because the neighbouring samples are moons

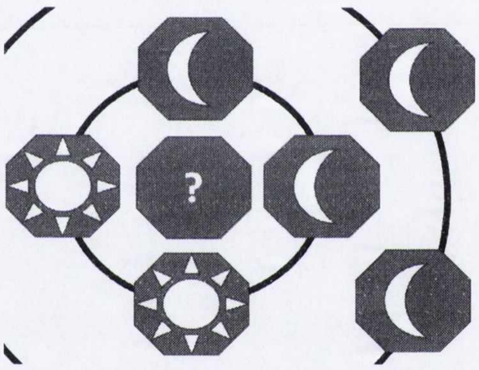
Figure 3.6: k-NN

3. The neighbour samples didn't vote. The SRL system is based on projecting annotation by alignment (see section 4.5), it can happen that a sub-tree is similar enough to be a neighbour but its alignment does not project a label. Figure 3.9 illustrates this phenomenon.



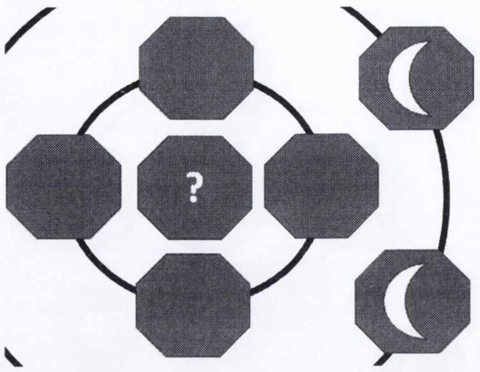
for k = 1, there are four first neighbours offering two different labels

Figure 3.7: Multiple neighbours at the same distance



The first nearest neighbours, which are the first equivalence class, produce a draw in the voting.

Figure 3.8: Draw in voting



The first neighbours did not vote because their alignment did not project a semantic label. If the system is forced to produce a prediction the parameter k has to be extended

Figure 3.9: Invalid voting

In the case of having multiple neighbours at the same distance, the policy of the 'nearest cut-off' was adopted. The parameter 'k' will be reduced or extended to the nearest value guaranteeing that all neighbours that are at equal distance receive the same treatment.

There are multiple ways to weigh the voting, for example making it proportionally inverse of the distance. The research about different possible ways to weight the voting is outside the scope of this thesis, and the impact of other voting methods on the tree-SRL system is left as an open question for future research. The policy adopted is that every vote has the same weight independent of its distance. In the case of a draw, the value 'k' will be extended.

3.4.2 Equivalence class

An equivalence class is a set of neighbour samples at equal distance to a reference sample (query). Figure 3.10 illustrates a sample where the query is represented with a "?" symbol and the neighbours with a half-moon. The horizontal line represents distance, and the vertical line groups the samples which are at the same distance. In this example there are three equivalence classes. The first equivalence class contains one sample, the second equivalence class contains three samples and the third equivalence class contains four samples.

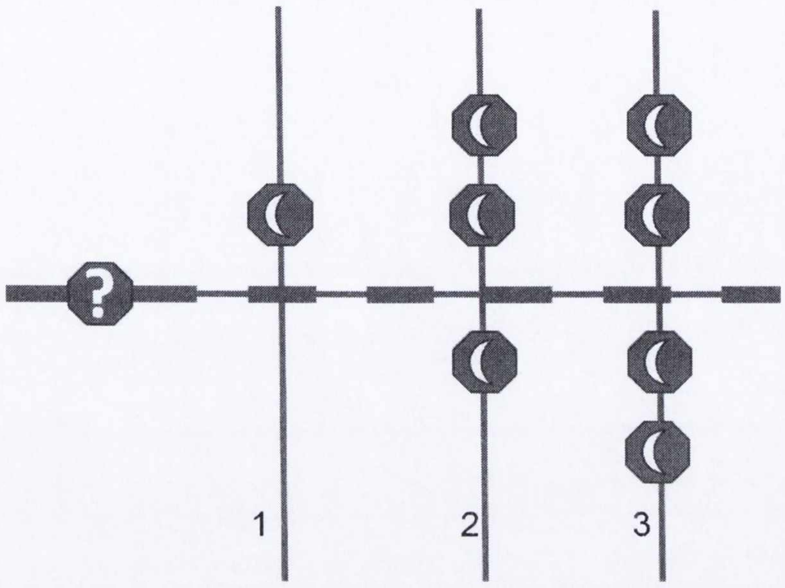


Figure 3.10: Equivalence class

The concept of equivalence class is important because k-NN algorithm works with equivalence classes and not with 'k' neighbours.

- The value k= 1, matches with the first equivalence class.
- The value k= 4, matches with the second equivalence class because it contains all samples on the first and second equivalence class.
- The value k= 3, does not match an equivalence class. Therefore the k value has to be extended to 4 or retracted to 1.

As the second equivalence class contains three neighbours at the same distance from the query, the algorithm will extend the value of k to four, in order to reach the nearest equivalent class. For this particular example the three equivalent classes correspond to k values of: 1, 4 and 8. Other values of k will have to shrink or be extended to fit them.

3.4.3 Linguistic example

The intention of using suns and moons was to explain how k-NN works without requiring knowledge from any other part of the thesis, this section attempts to illustrate the same concepts but in the context of the Tree-SRL system.

Assuming it is required to annotate the following sentence:

(31) [John]$_?$ **ran** [from home]$_?$ [to the university]$_?$.

And the following sentences were found in the first equivalence class:

(32) [Yesterday,]$_{Temp}$ Peter **ran** [to my house]$_{Goal}$.

(33) The **run** [from the station]$_{Source}$ was [exhausting]$_{Manner}$.

(34) [Paul and Mary]$_{Runner}$ **ran** [yesterday]$_{Temp}$.

Table 3.3 shows a possible alignment from the annotated sentences to the query sentence.

Table 3.3: Example of alignments

Query sentence	sample 32	sample 33	sample 34
	Yesterday,$_{Temp}$		
John$_?$	Peter$_{Runner}$	The	Mary$_{Runner}$
			and
			Paul
ran	**ran**	**run**	**ran**
from$_?$		from $_{Source}$	yesterday$_{Temp}$
		the	
home		station	
to$_?$	to$_{Goal}$	was	
the	my		
university	house	exhausting$_{Manner}$	

The Tree-SRL system will use tree representations but in order to simplify the explanation the alignments are given on string representation.

In this way the query sentence can be seen as a vector of three unknown semantic labels, and it is possible to have one k-NN panel for each of them. Looking just to the relation of the argument "from home" to the predicate run, sentence 32 would project nothing (null vote), sentence 33 would project "Source", and sentence 34 would project "Temp", the k-NN module would detect a drawn in voting (like in Figure 3.8) and will try to extend the panel of alignments.

Looking just to the relation of the argument "John", to the predicate run, sentence 32 would project "Runner", sentence 33 would project nothing (null vote), and 34 would project "Runner", so the k-NN module will predict "Runner" as the semantic relation between John and the predicate run (in Sentence 31). If sentence 33 was the complete first equivalence class and sentences 32 and 34 the second equivalence class, the case would be like the one in Figure 3.7, where the panel has to be extended because no valid votes are found in the first equivalence class.

3.4.4 k-NN in SRL

Li et al. (2009) were the only participant group in the four CoNLL evaluations for SRL who uses k-NN algorithm. They used k-NN only for the sense disambiguation subtask of the predicates. Argument identification and classification task were done through maximum entropy. k-NN was used for predicate sense disambiguation but not in all languages, for Chinese, Spanish and Catalan SVM was used. They compared the results using SVM and k-NN for German and concluded that 20-NN performs better on that particular data set. English and Czech present a different scenario; these data sets are too large. Therefore, they decided to use 20-NN for English and 10-NN for Czech because SVM was 'unacceptably slow' to process the data within the time limits of the public evaluation.

SVM requires a substantial computational time investment in the learning phase which depends on the size of the training data set. Since k-NN is a 'lazy-learning algorithm'; it requires a substantial computational time investment in the labelling phase. Its computational cost depends on the size of the training data set by the size of the evaluation data set.

Li et al. (2009) found a way to avoid the huge computational cost. They used as many k-NN classifiers as different predicate frames (same sense) can be found in the training data set, which reduces the possible neighbours substantially.

Therefore, the computational cost was reduced to the average size of an amount of samples of each frame by the size of the evaluation data set.

This strategy of ignoring samples of different predicate seems reasonable because there is no strong intuitive reason to assume that the meaning of different predicates are related to each other by their tag numbers.

3.5 Conclusion

This chapter introduced the basic concepts required to understand the system and how it will be evaluated. It starts by explaining that for argument classification task only, F1 measure gives identical values to accuracy values, so consequently accuracy was adopted. It also introduces the McNemar test which is suitable to evaluate variants of a classifier making predictions on the same data set (but not making predictions on a different data set). It introduced the tree edit distance algorithm which produces Tai-mappings between pairs of trees and also calculate a distance scores, detailed how the mappings and scores are calculated and what are their restrictions. Finally, the chapter ends by explaining the k-NN algorithm with a minor adaptation to deal with samples that will not produce any label.

The next chapter will explain the architecture of the proposed SRL system and will make use of the concepts explained in this chapter.

Chapter 4

Tree-SRL

4.1 Introduction

The Tree-SRl is the main contribution of this thesis. To explain how this works, it is necessary to describe how the sub-trees are extracted from the original data which are the basic sample units, and how an alignment between two sub-trees can be used to project the labels of one onto the other, which is the first part of the chapter. The second part describes how to deal with a list of alignment sorted by its distance to the sub-tree to be labelled and how k-NN works. In further sections the atomic distances between nodes of sub-trees that were used in the experiments are described. The chapter finishes by explaining a similar system develop by Furstenau and Lapata (2011).

4.2 Sub-trees

The Tree-SRL system works with sub-trees; each sub-tree contains a single predicate node and can contain several argument nodes (Figure 4.1), which is probably the most intuitive concept of sub-tree. This section describes a range of possible options to extract those sub-trees.

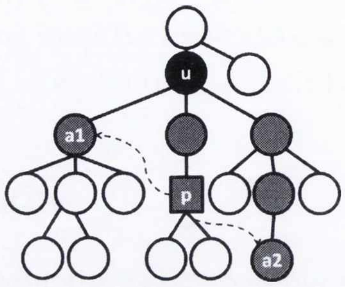


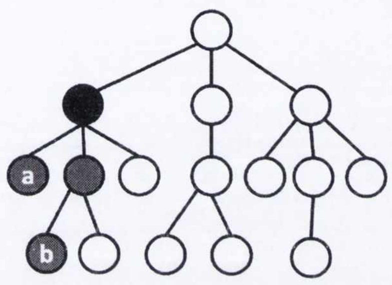
Figure 4.1: First sample of sub-tree extraction

```
Input: T:tree structure labelled in post order traversal
Input: L:list of nodes to be labelled in post order traversal
Output: T:Sub-Tree
foreach node x in the list do
    mark x as part of the sub-tree;
end
while the list contains no uniform values do
    [minValue, position] = min(L);
    value = parent(minValue);
    mark value as part of the sub-tree;
    L[position]=value;
end
Remove all nodes that are not marked as part of the sub-tree;
Figure 4.2 illustrates the strategy
```

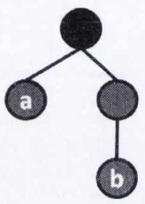
Algorithm 2: Sub-tree extraction

Figure 4.1 illustrates the process and Figure 4.2 shows two samples. Assuming that 'p'(the square node) is a predicate node and nodes 'a1' and 'a2' are its arguments [1] (the arguments are defined by the semantic relationships, in this case the semi-dotted arrows.), the sub tree extracted from the above sentence will contain the nodes 'a1', 'a2', 'p' and all ancestors of 'a1', 'a2' and 'p' up to the first common one (in this case node 'u'), which is also included in the aforementioned sub-tree. None of the white nodes are included in the sub-tree. The straight lines represent syntactic dependency relationships.

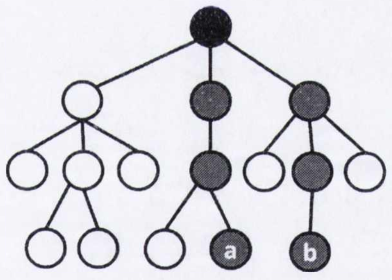
[1]'p','a1' and 'a2' are nodes assumed to be already identified.



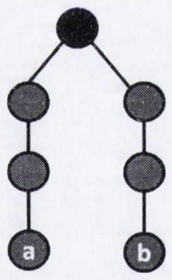
(a) sample tree



(b) sub-tree sample 1



(c) sample tree



(d) sub-tree sample 2

The minimal sub-tree for a set of nodes is the one which contains the paths between the nodes selected to be in the sub-tree (in these samples node 'a' and node 'b') through the nearest common ancestor. The nearest common ancestor is drew in black and the paths are drew in grey. All dark nodes belong to the sub-tree. Algorithm 2 describes this implementation.

Figure 4.2: Two samples of sub-tree extraction

4.3 The labelling sub task

```
Input: training data set (labelled)
Input: testing data set (unlabelled)
Output: testing data set (labelled)
load training and testing data;
adapt the trees for the tree distance algorithm;
foreach sentence from training and testing data do
    obtain each minimal sub-tree for each predicate;
end
foreach sub-tree T from the testing data do
    calculate distance and the alignment from T to each training sub-tree;
    sort the list of alignments by distance;
    use the list to label the sub-tree T ;
end
```

Algorithm 3: Tree-SRL system pseudo code

```
Input: dependency tree structures
Output: tree structures, unlabelled arrow
foreach tree do
    foreach arrow do
        move the label to the child node;
    end
    add label 'root' to the empty arrow label slot on the head node;
end
```

Algorithm 4: Tree adaptation for tree distance algorithm

The Tree-SRL system performs a sub task of the SRL-only closed challenge. For this task, predicates and their arguments are already identified. Therefore, the system only has to label the relationships. The predicate disambiguation task is ignored.

The Tree-SRL system is implemented in C++. The 'pseudo code' of the system is described in Algorithm 3, where:

- Trees are preprocessed in order to move the dependency labels from the arrows into node labels. (Algorithm 4)
- A minimal sub-tree is the one that contains at least the predicate and can also contain arguments.

- There are several ways to use the sorted list for labelling a sub-tree. They are discussed in section 4.6.
- The list is sorted by ascending tree distance order.

4.4 Adapting dependency structures

The tree distance algorithm is not designed for working with labelled arrows, but with labelled nodes[2], what makes necessary to change the trees into a new type without labelled arrows. Possible ways to do this are:

1. Remove the labels of the arrows. This strategy loses important information (the dependency relationships).
2. Add a new intermediate node between the nodes of the original tree which has a relationship represented by an arrow. This strategy nearly duplicates the amount of nodes and has the negative effect of having nodes with different meanings (some of them are words, some of them relationships).
3. Move the label of the arrow into the child node. This strategy has been implemented and described in Algorithm 4.

The tree distance implementation of the tree-SRL system, accepts multiple labels per node, and the cost of matching two nodes depends on how many different labels there are.

4.5 Projecting labels though mappings

Any kind of mapping with a one-to-one restriction can be used for projecting labels. In this semantic role labelling task the semantic relationships are between two single nodes; the predicate node and the argument node.

If a one-to-one partial mapping from the nodes of tree S into the nodes of three T, matches two predicate nodes $f(p^S) = p^T$,[3] and two of their corresponding argument nodes also match each other $f(a_x^S) = a_y^T$, then the semantic

[2]Section 2.4.1 gives some details about the format of the data on the CoNLL 2009 shared task.

[3]$f(a) = b$: means that the mapping matches node a to node b.

relationship between nodes p^S and a_x^S can be projected as a relationship between p^T and a_y^T.[4]

Figure 4.3 illustrates this process with samples from the English data set.

Figure 4.3 shows two sentences in a dependency tree representation, One is labelled (the training sentence, bottom side) and the other is unlabelled (the testing sample, top side). At each node the word form and its position in the sentence is shown.

Straight arrows represent syntactic dependencies. The square node represents the predicate that is going to be analysed (There can be multiple predicates in a single sentence, but in this sample only the predicate "*rose*" is shown). Dashed arrows between a square node and an elliptic node represent a semantic relationship. These arrows have a semantic tag: for the labelled sentence (A1, A2 and A4). The alignment predicts the labels A1, A2 and A4 (producing two correct predictions and one wrong, thus the correct labelling is A1, A2 and A3).

The sub-tree which is labelled is called the source sub-tree and the one in which the projection is made is called the target sub-tree. If the mapping used does not produce the matches needed to project the label on the target sub-tree then the projection is invalid. In the case of using multiple projections to vote what label should be predicted (which is the case of using k-NN, explained in section 3.4) invalid projections will count as null votes.

4.6 Strategies to make predictions

There are multiple ways to use k-NN to select "n" sets of neighbors labelled samples to label "n" pre-identified arguments for a single predicate. This section details three different algorithms to do it which are implemented and mention briefly another two possible strategies to do it which are not implemented. No further strategies are mention because I could not imagine any other way to do it.

[4]Note that not all semantic relationships of T have to be predicted by S, the alignment can predict just part of the annotation.

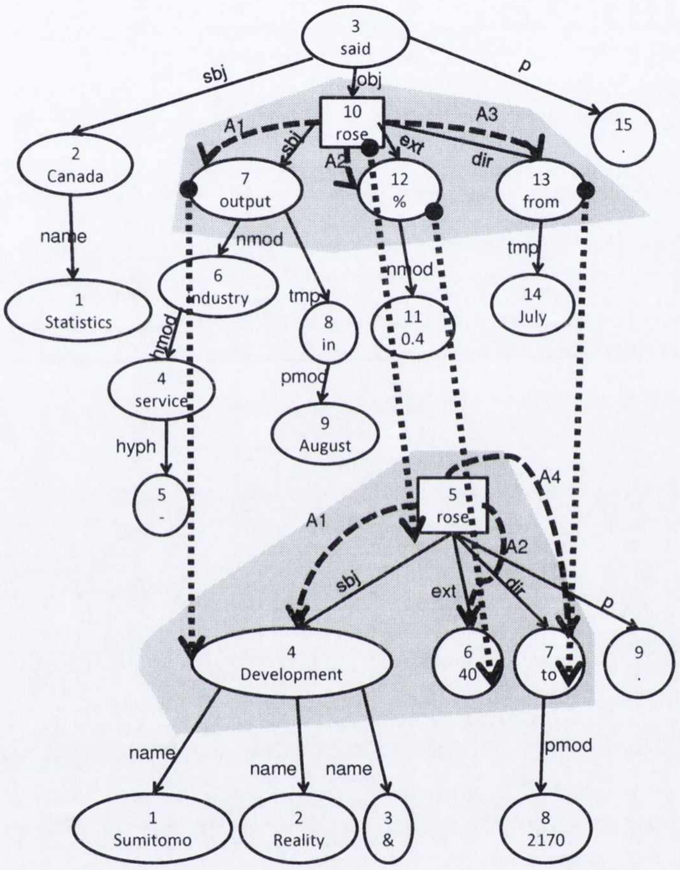


Figure 4.3: The projecting label process
For clarity: dependency relations are drew at the arrows of the dependency in stead of inside the dependant node.

4.6.1 One argument at a time

```
Input: sub-tree to be labelled
Input: list of mappings sorted by ascending tree distance
Output: label for the relationship
foreach predicate (p) in T do
    foreach argument (a) dependent on p do
        foreach equivalence class EC in the sorted list do
            foreach mapping (map) in EC do
                if $|sr(f_{map}(p), f_{map}(a))| > 0$ then
                    knnPanel $\leftarrow sr(f_{map}(p), f_{map}(a))$;
                end
            end
            if knnPanel is ready then
                break loop;
            end
        end
        $sr(p, a) \leftarrow knnPanel.getPrediction()$ ;
    end
end
```

- 'p' is the node predicate in T.
- 'a' is a node argument in T.
- 'map' is a mapping between the sub-tree that has to be labelled and a sub-tree in the training data set.
- $sr(p, a)$ gives the set of semantic labels of the relationship between the predicate node 'p' and the argument node 'a'. $|sr(x, y)| > 0$ is true if there is at least one semantic relationship from predicate node x and argument node y.
- $f_{map}(x)$ gives the matched/image node of the node x through the maping 'map'. The method function is explained in Figure 3.1 on page 42.
- *knnPanel* is ready when the panel contains at least one sample and there are no ties on the voting.
- *knnPanel.getPrediction*() is a method that return the set of labels who win the voting in the k-NN panel.

Algorithm 5: One-at-a-time strategy

Algorithm 5 gives a pseudo code for the case that the requested $k = 1$, and accordingly the attempts to make a prediction starts with the first equivalence class.

This strategy produces an independent k-NN panel for each of the predicates to be labelled, which may contain different amount of equivalence classes for each argument. The system predicts argument by argument without keeping any information used for the previous argument of the same predicate.

In general for $k > 1$, a prefix 'PRE', of the sequence of equivalence classes is found whose size is closest to k, and the attempt to make a prediction starts with the equivalence classes with this prefix. Accordingly, the pseudo-code may be seen as describing the behaviour with $k > 1$ if the first equivalence class is thought of as the above described prefix.

4.6.2 One-at-a-time ignoring equivalence classes

This is a naive simplified version of one-at-a-time strategy which assumes that each equivalence class contains a single example, in fact the strategy consist on ignoring the existence of equivalence classes which contain more than one sample. The version for $k = 1$ does not require an elaborated k-NN implementation because there is no panel, no voting, and so, there are no voting ties. Algorithm 6, shows the pseudo code. The purpose of experimenting with this algorithm is to analyse the impact of the k-NN module on the system and the awareness that equivalence classes usually contain more than a single sample.

```
Input: sub-tree to be labelled
Input: list of mappings sorted by ascending tree distance
Input: k value
Output: label for the relationship
foreach predicate (p) in T do
    foreach argument (a) dependent on p do
        foreach mapping (map) in the sorted vector do
            k=k-1;
            if $|sr(f_{map}(p), f_{map}(a))| > 0$ then
                knnPanel $\leftarrow sr(f_{map}(p), f_{map}(a))$;
                if knnPanel is ready and k<1 then
                    $sr(p, a) \leftarrow knnPanel.getPrediction()$ ;
                    break loop;
                end
            end
        end
    end
end
```
Algorithm 6: One-at-a-time strategy ignoring equivalence classes

4.6.3 All arguments at a time

One-argument-at-a-time permits the system to use a different k-NN panel size to predict each of the arguments of the same predicate.

An alternative way could be to force the system to use the same panel of samples to predict each of the arguments of a single predicate. Note that the same panel of samples will produce a different panel of votes for each argument of the same predicate. This happens because the mapping of a single sample will vote or project different label for the different semantic relationships on the same sub-tree.

In this way, all neighbours samples used have to vote for all semantic relationships that have to be predicted, and the prediction of the system is only ready when all panels are ready to produce a prediction. Algorithm 7 details the process.

```
Input: sub-tree to be labelled
Input: list of mappings sorted by ascending tree distance
Output: label for the relationship
foreach predicate (p) in T do
    foreach equivalence class EC in the sorted list do
        foreach mapping (map) in EC do
            foreach argument ($a_i$) relative to p do
                if $|sr(f_{map}(p), f_{map}(a))| > 0$ then
                    knnPanel[i].add($sr(f_{map}(p), f_{map}(a_i))$);
                end
            end
            ready = true;
            foreach argument ($a_i$) relative to p do
                if knnPanel[i] is NOT ready then
                    $ready = false$ ;
                end
            end
            if ready==true then
                break loop ;
            end
        end
    end
    foreach argument ($a_i$) relative to p do
        sr(p,a$_i$) $\leftarrow knnPanel[i].getPrediction()$;
    end
end
```

- with this strategy for each predicate there are as many k-NN panels as argument nodes.

Algorithm 7: All-at-a-time strategy, k=1

4.6.4 Further Strategies to make predictions

The following strategies were not implemented.

4.6.4.1 Golden samples

Another alternative would be to ignore any sample which can not produce a valid vote for all semantic relationships to be predicted.

4.6.4.2 Threshold mappings

Another alternative would be to ignore any sample which does not produce a certain percentage of valid votes for all semantic relationships to be predicted.

It is more flexible than 'golden samples' because in some cases it can be very difficult to find samples which can produce valid votes for all arguments.

4.7 Atomic settings

The atomic settings used on the tree distance system have an important impact on the performance of the system: it not only modifies the ranking of neighbours by distance, it can also modify the mapping between the nodes of the two taken sub-trees.

There is an infinite range of possibilities for calibrating the atomic costs. This thesis explores the effects of four different atomic settings: Ternary (based only on syntactic information), Hamming (extending ternary with lexical information) and their frame match versions (in which swapping predicates nodes of a different frame have an extra cost of one). Two other systems were used as baselines: Shape and Binary

Shape

- Swaps have a cost of 1.

Figure 4.4 shows the mapping sample for Shape settings for the example in Figure 4.3 on page 63.

Binary

Binary settings are named in this way because the atomic distance between two nodes can only be zero or one.

Swaps cost:

- 1 if POS or DepRel are different
- 0 otherwise.

Figure 4.5 shows the same two trees in the way that the Binary system would see it (The Ternary Binary system concatenates POS and DepRel as a label for the node.

Ternary

The swap cost is the sum of the following costs:

- +0.5 if POS is different;
- +0.5 if DepRel is different;

It is named "Ternary" because there are three possible swapping cost outcomes: 0, 0.5 and 1.

These costs are chosen to produce a smoother distance measure improving the Binary system. Figure 4.6 shows the mapping sample for Ternary settings for the example in Figure 4.3 on page 63.

Hamming

This settings is call Hamming because it gives equal weight to each label mismatch. The swap cost is the sum of the following costs:

- +0.25 if POS is different;
- +0.25 if DepRel is different;
- +0.25 if Lemma is different;
- +0.25 if Form is different;

This atomic cost setting is designed to use all information available. Figure 4.7 shows the mapping sample for Hamming settings for the example in Figure 4.3 on page 63.

Frame Match versions

The Frame Match versions of the atomic costs present an extra cost of one unit for swapping two predicate nodes from different frames.

For **Frame Ternary** the swap cost is the sum of the following costs:

- +0.5 if POS is different;
- +0.5 if DepRel is different;
- +1 if one node is a predicate and the other is not or if both are predicates which belong to a different frame.

For **Frame Hamming** the swap cost is the sum of the following costs:

- +0.25 if POS is different;
- +0.25 if DepRel is different;
- +0.25 if Lemma is different;
- +0.25 if Form is different;
- +1 if one node is a predicate and the other is not or if both are predicates which belong to a different frame.

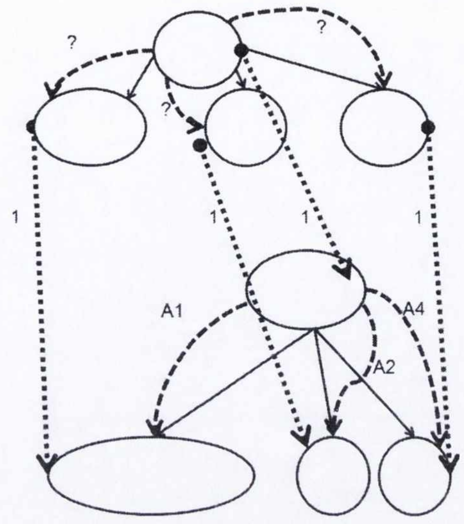
Figures 4.8and 4.9 shows the mapping sample for Frame Ternary and Frame Hamming settings for the example in Figure 4.3 on page 63.

Insertion and deletion cost

In all experiments insertion and deletion cost is equal to one unit. What is variable is the swapping cost.

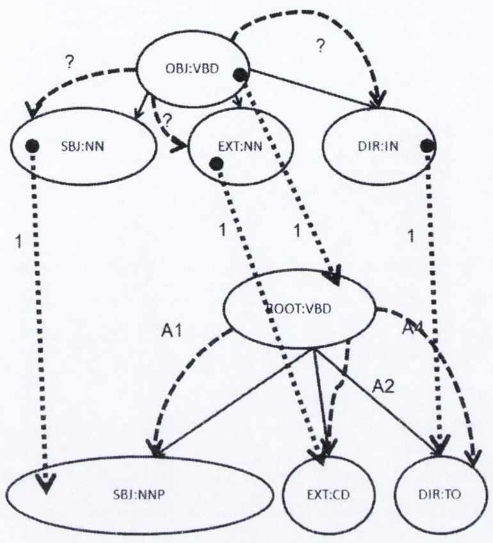
Illustrations of the atomic settings

This section illustrates how Figure 4.3 on page 63 looks for the system when using each of the different atomic settings previously described. The numbers in the swapping arrows represent the cost of that particular atomic swapping.



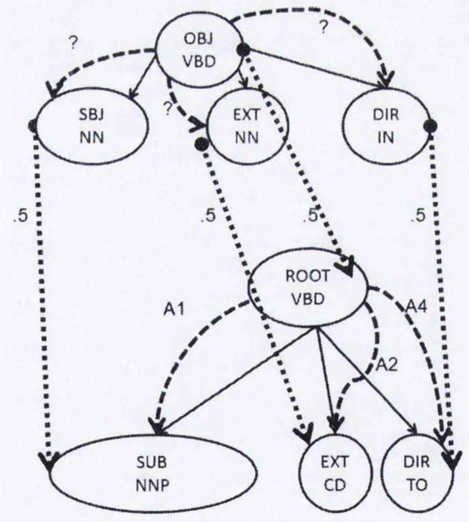
distance=4

Figure 4.4: Shape settings



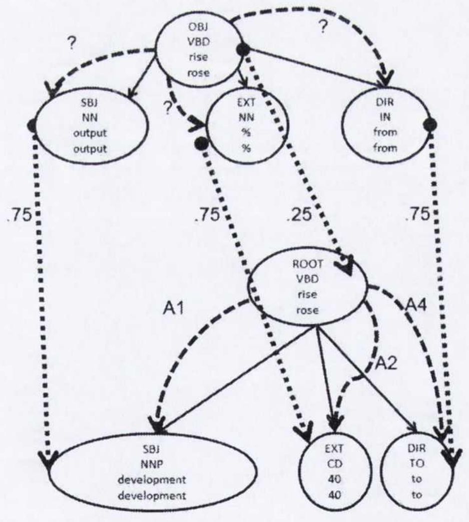
distance=4

Figure 4.5: Binary settings



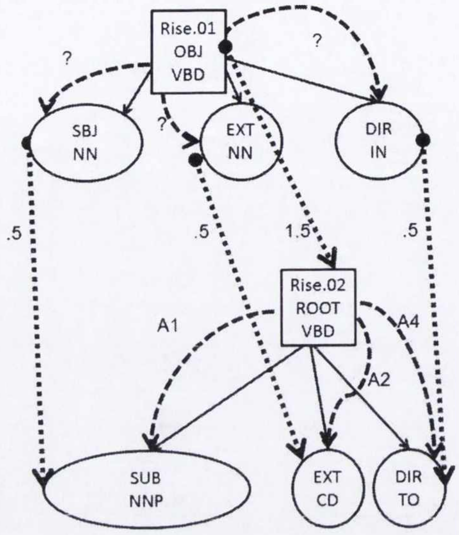
distance=2

Figure 4.6: Ternary settings



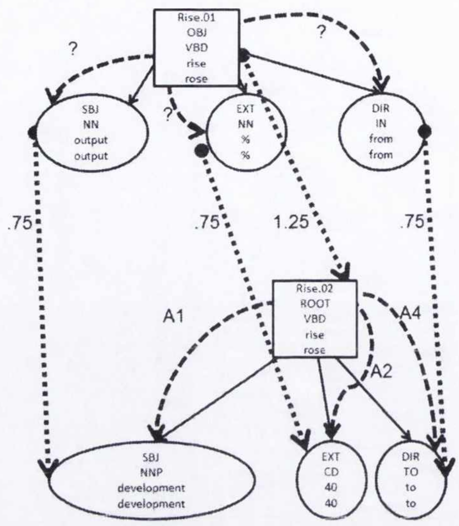
distance=2.5

Figure 4.7: Hamming settings



distance=3

Figure 4.8: Frame Ternary settings



distance=3.5

Figure 4.9: Frame Hamming settings

4.7.1 Expected performance

The atomic settings are designed by increasing complexity, and it is expected that the more complex settings perform better than the simpler versions. Therefore the following results are expected:

H1 Shape < Binary

H2 Binary < Ternary

H3 Ternary < Hamming

H4 Ternary < Frame Ternary

H5 Hamming < Frame Hamming

H6 Frame Ternary < Frame Hamming

A<B means that the accuracy of the system using the A setting is lower than the accuracy of the system using the B settings.

4.7.2 Future work

The possibility of considering smother similarities between nodes could include smother similarities between words: synonyms may have lower swapping cost and smother similarities between dependency relations, for instance NN and NNP relations are quite similar.

4.8 Important consideration on data set

Due to the excessive computational cost of the k-NN algorithm the training data sets of Czech and English were reduced by loading only the first 10,000 sentences for Czech and the first 20,000 sentences in English.

This decision is a systematic under-sampling method, which is not necessarily the optimal to preserve the performance of the system and can add the risk of reducing the scope of the domain, by picking all the samples of one sub-topic and ignoring all the samples of another sub-topic. On the positive side, this decision should make the experiments easier to replicate with more precision than what would be expected by random under-sampling. Sophisticated under-sampling methods may increase the accuracy but they are outside the scope of this thesis.

4.9 The k-NN choice

The system is not using k-NN to classify semantic structures or arguments, but to select a panel of samples. Each of those samples may contain annotation for multiple predicate-argument relations. Those relations will be projected into unlabelled query as a prediction.

In order to project a label it was necessary to have a one-to-one alignment between a labelled sub-tree and the unlabelled sub-tree. Tree edit distance algorithm produces a one-to-one alignment and a distance score.

Alternative algorithms such Tree Kernels (Moschitti et al., 2008) can compare tree structures but do not create the alignment need it for the projection. Tree Kernels can be use for classification, but not to project alignment. Therefore k-NN was chosen as the machine learning algorithm.

4.10 Similar systems

As far was the author of this thesis is are aware, there has been little work exploring an alignment approach to SRL, an exception being Furstenau and Lapata (2011). In their work they also consider projection of annotation from within a labelled dependency tree to an unlabelled one, but there are many points at which their work differs from that presented in this thesis. They work with the FrameNet annotation inventory (Baker et al., 1998). They also align tree by applying a a graph-edit alignment notion, one which does not pay any attention to ancestry or linear order. The alignment score they use is similarity-based, and refers to lemma and dependency relationships. They do not consider alternatives, such a distance-based scoring. A further difference is that their aim is not per-se to annotate an unlabelled *corpus* in its entirety, but rather from amongst generated annotated versions of the unlabelled data (the *expansion* corpus in their terminology) to select cases to add as exemplars to a frame *lexicon* (the *seed* corpus in their terminology). Rather than basing the annotation of an unlabelled item, T, on its nearest labelled neighbours, $\{S_1, \ldots, S_n\}$, they take each *seed* item, S, in turn and use it to project annotation to its nearest neighbours in the expansion corpus, $\{T_1, \ldots T_n\}$. For all of these reasons, it was not possible to make any meaningful quantitative comparisons with their work. Nonetheless, it seems reasonable to expect the contrasts already described concerning cost settings and distance versus similarity to apply to the kind of data-set expansion scenario they discuss and investigating whether this is so is a potential avenue for further research. Conversely it would be interesting to see how the findings are affected if the notion of alignments were replaced, which must be Tai mappings, with the notion of alignment from their work, which is left for future work.

4.11 Conclusion

This chapter explained the architecture of the Tree-SRL system, which performs the argument classification sub-task of a Semantic Role Labelling system. It started by introducing how the system extracts sub-trees containing a predicate with all its arguments from the dependency tree representation of the text. The alignments provided by the tree edit distance algorithm allow the system to transfer the labels from one annotated sub-tree to another one as a prediction, and the distance score to rank the alignment, which enables a k-NN module to select which of the alignments will be used on the prediction. There is more than one way to use the list of ranked alignments and the system implements three of them,with their impacts to be analysed in Chapter 8.

The alignments and the tree distance score depends on the atomic costs for swapping, deleting or inserting nodes. This chapter proposes four atomic settings plus another two as base lines. For all atomic settings proposed the deletion and insertion cost is fixed to one unit. There are two base lines: Shape and Binary settings. The Shape setting assigns a cost of one unit to any swap independent of the content of the nodes being swept. The Binary setting assigns a cost of one or zero depending on the syntactic feature. The non-base line measures are Ternary, uses syntactic features and can produce the cost values, and Hamming,uses the lexical features producing up to five different cost values. For both of these, there is a Frame Match version in which there is extra cost for swapping predicate nodes which belong to a different semantic frame. The expectation is that with the increasing complexity of the atomic cost settings the accuracy of the system will also increase. this would be tested in Chapter 5. Regarding the parameter k of the k-NN module the default adopted value is one. This decision will be tested in Section 8.4 on page 203. The chapter ends referencing a similar system to the one proposed in this thesis.

The description of the source code, sanity tests, user manual and computational cost analysis. are placed in Appendix B. The source code can be download from https://github.com/francoph/Tree-SRL.git

Chapter 5

Distance outcomes

5.1 Introduction

This chapter presents the main results of the experiments, compares and comments on them in detail. It starts with describing the baselines, and comparing the atomic measures to each other. Then it continues explaining details of the observation and finishes by contrasting the impact of using tree structures against using linear structure.

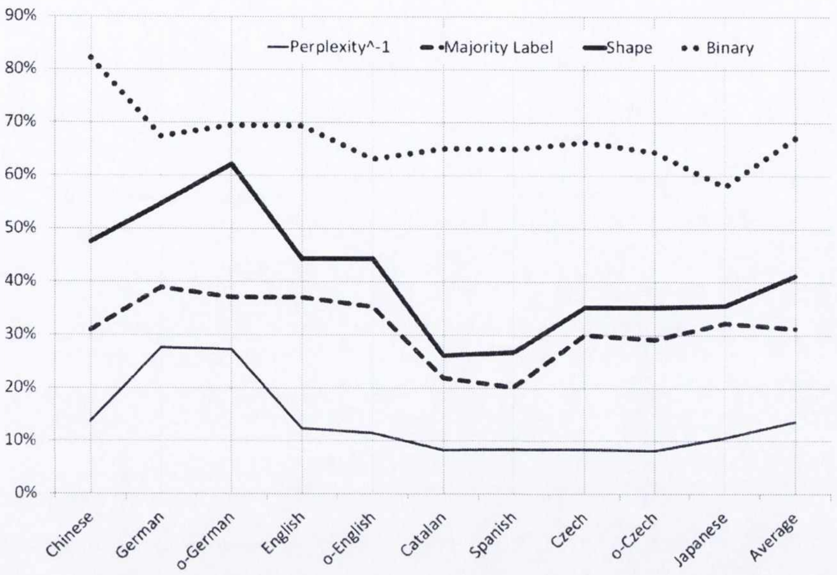
5.2 The base lines and inverse perplexity

Figure 5.1 shows the inverse perplexity of the set of semantic roles and the three base lines: Majority Label, Shape, and Binary settings.

In a text where all words or events are equally probable, the perplexity would be exactly the same as the size of the alphabet. Therefore, if the probabilities of the semantic labels would be balanced the majority label settings would be as accurate as the value of the inverse of perplexity. The frequency of the semantic labels is very imbalanced and that is why there is a large margin of difference between the inverse perplexity and the accuracy of the Majority Label.

All atomic measures settings including Shape and Binary were defined in Section 4.7, page 68. It was unexpected[1] that Shape (settings) performs better than Majority Label for all data sets.

[1]The author had an intuition that both would score similarly to each other



Percentages	*Perplexity*$^{-1}$	Majority Label	Shape	Binary
Chinese	13.65	30.86	47.49	82.14
German	27.59	38.96	54.66	67.41
o-German	27.27	37.05	62.06	69.51
English	12.35	37.06	44.24	69.34
o-English	11.41	35.33	44.27	63.08
Catalan	8.26	21.90	26.19	65.12
Spanish	8.33	20.17	26.73	64.97
Czech	8.32	29.98	35.17	66.28
o-Czech	8.15	29.08	35.18	64.42
Japanese	10.51	32.21	35.47	57.93
Average	13.58	31.26	41.15	67.02

Figure 5.1: Accuracy across the four Base lines and inverse perplexity of the distribution of labels

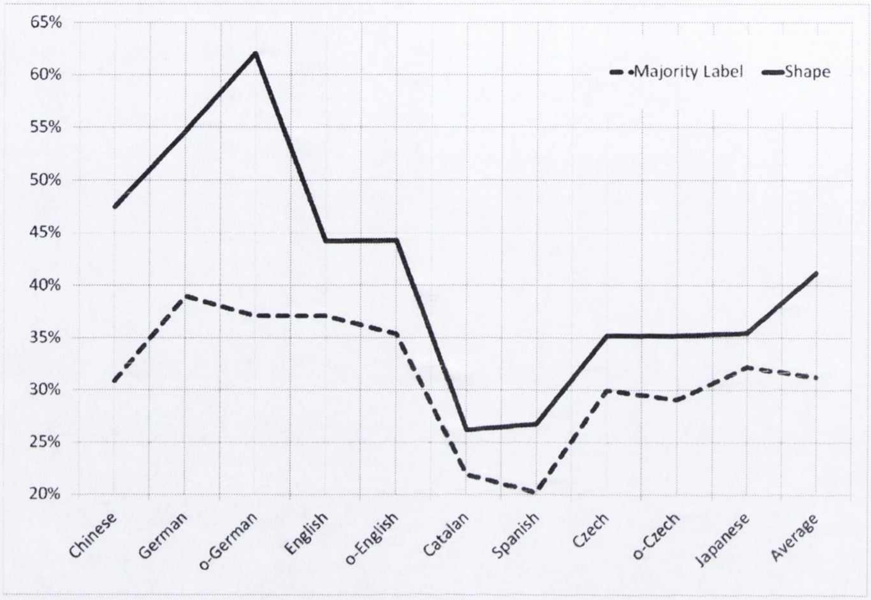
The semantic roles of the German language[2] showed the highest inverse perplexity (over 27%) which is probably due to having the smallest alphabet: 10 labels where the average for all ten data sets is 47.

Another observation is that the majority label base line and the inverse perplexity in German in-domain and out-of-domain get closer than in any other data set, but the Shape setting in out-of-domain performs better than in-domain.

The four lines do not always follow each other, for instance Spanish and Catalan show a valley for Shape and Majority label but not for Binary and inverse perplexity.

[2]The set of roles can be seen as classes of a classification task.

The Shape setting uses only the structural information and always performs better than the Majority Label base line, suggesting that structure information is very useful for classifying arguments. Figure 5.2 illustrates the comparison in a isolated way. Section 5.3.5 will give more details about Shape.



Percentages	Majority Label	Shape
Chinese	30.86	47.49
German	38.96	54.66
o-German	37.05	62.06
English	37.06	44.24
o-English	35.33	44.27
Catalan	21.90	26.19
Spanish	20.17	26.73
Czech	29.98	35.17
o-Czech	29.08	35.18
Japanese	32.21	35.47
Average	31.26	41.15

Figure 5.2: Accuracy Majority Label vs Shape setting

5.3 Contrasting swapping costs

This section is going to describe the main aspects of the results all together and after that it will turn into the further details of those aspects, its analysis and possible explanation for why the results are in such way.

5.3.1 Main observations

Figure 5.3 shows a graph with the accuracy for the seven languages[3] on the evaluation data set including the three out-of-domain cases, along with three base lines and the Table below the graph shows the exact values.

The in-domain languages are sorted by the average accuracy of the four atomic measures followed by the out-of-domain results.

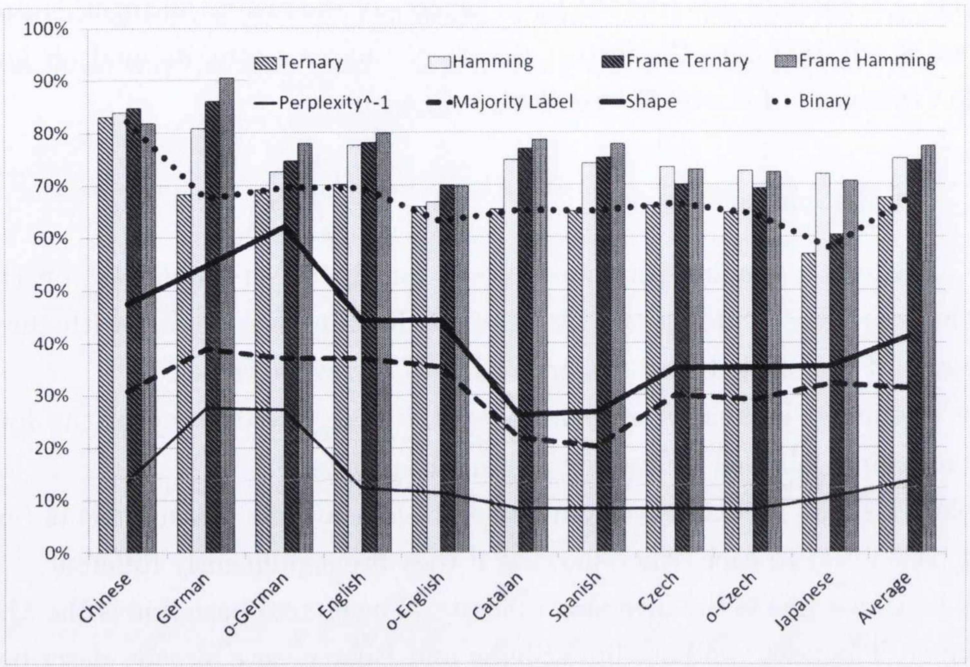
Table 5.1 shows an extensive comparison of four atomic measurements (excluding base lines) to each other showing if they are significantly different.

The first base line is the inverse perplexity. The second base line is the Majority label. The next two base lines Shape and Binary were already described in Section 4.7 at page 68 and discussed in Section 5.2.

All four atomic measures clearly out-perform Majority Label and Shape. And on average across all languages, "Frame Hamming" achieves the higher performance (see Figure 5.3 average column),

Table 5.1 compares extensively all four settings, and can be read as follows: (T) for Ternary settings, (H) Hamming, (FT) Frame Ternary and (FH) Frame Hamming. Each line of the table corresponds to one language, where the first one is Chinese. The first line of the Chinese comparisons: starts with "T-H" means that Ternary was compared with Hamming. Ternary (at the column 'first') produces an accuracy of 83% and Hamming (at the column 'second') produces an accuracy of 83.93%. Results are given with two decimals. The last column of the table shows the difference between the first and second columns. At the right side of the comparison two stars appear meaning that the McNemar test detected a significant difference at a strict significant threshold (p=0.001), the other options are one star for significantly different (p=0.05) or an equal symbol to indicate that no significant difference was detected. The exclamation

[3]For deletion/insertion cost equal to one.



Percentages	*Perple–xity*$^{-1}$	Majority Label	Shap	Bina	Tern	Hamm	Fram Tern	Fram Hamm
Chinese	13.65	30.86	47.49	82.14	83.00	83.93	84.78	81.79
German	27.59	38.96	54.66	67.41	68.25	80.82	86.03	90.50
o-German	27.27	37.05	62.06	69.51	69.10	72.53	74.62	77.89
English	12.35	37.06	44.24	69.34	70.12	77.57	77.98	79.86
o-English	11.41	35.33	44.27	63.08	65.77	66.68	69.83	69.69
Catalan	8.26	21.90	26.19	65.12	65.23	74.76	76.77	78.51
Spanish	8.33	20.17	26.73	64.97	64.89	73.95	75.04	77.61
Czech	8.32	29.98	35.17	66.28	65.84	73.28	69.99	72.74
o-Czech	8.15	29.08	35.18	64.42	64.73	72.56	69.24	72.23
Japanese	10.51	32.21	35.47	57.93	56.69	71.92	60.43	70.53
Average	13.58	31.26	41.15	67.02	67.36	74.80	74.17	77.43

Seven languages and three out of domain data sets
The inverse perplexity of the semantic labels and three base lines was also drawn: (1) the prediction of a system that always predicts the majority label on the training data set, (2) a tree system blind to any label (Shape) where the swapping cost, insertion and deletion is always 1 and (3) a Binary system where the swapping cost is 1 if DepRel or POS labels are different, this base line is almost as good as the Ternary system
The in-domain languages are sorted by the average accuracy of the four bars.

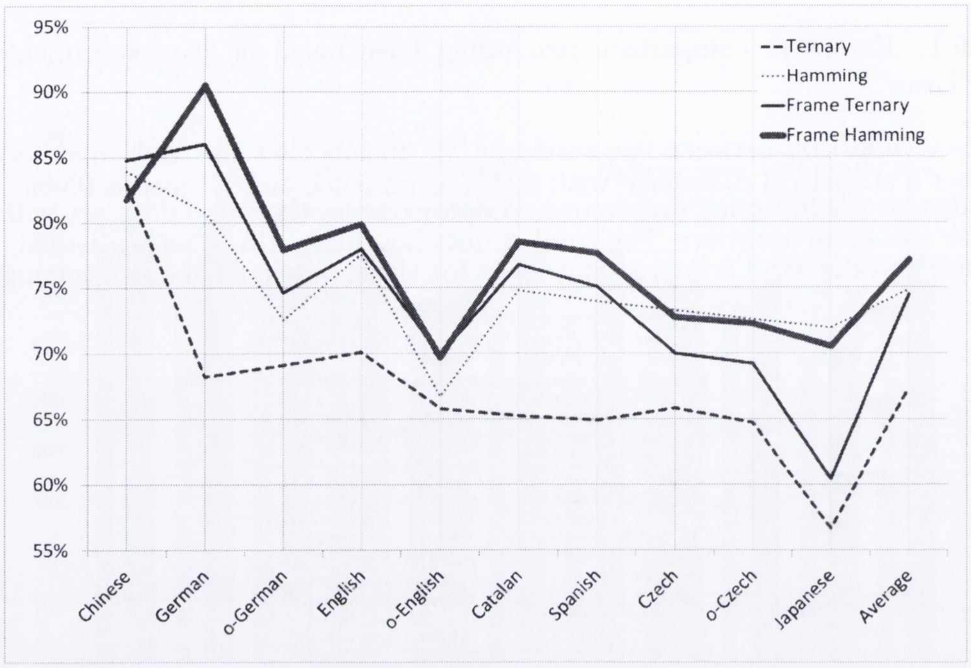
Figure 5.3: Accuracy of Tree edit distance

Table 5.1: Extensive comparison (excluding base lines) of the experimental atomic costs

On the comparison between two settings: '**' means that the McNemar test detect a statistical difference with 99.9% confidence and '*' means 95 % confidence. All following one to one comparisons on the same data set will follow the same notation. The total counts hoy many times an statistical difference was detected across all data sets for those particular pair of settings.

	accuracy	first	second	difference
Chinese	T-H **	83%	**83.93%!**	0.9238%
	T-FT **	83%	**84.78%!**	1.779%
	T-FH **	**83%!**	81.79%	-1.216%
	H-FT **	83.93%	**84.78%!**	0.8552%
	H-FH **	**83.93%!**	81.79%	-2.14%
	FT-FH **	**84.78%!**	81.79%	-2.995%
German	T-H **	68.25%	**80.82%!**	12.57%
	T-FT **	68.25%	**86.03%!**	17.78%
	T-FH **	68.25%	**90.5%!**	22.25%
	H-FT **	80.82%	**86.03%!**	5.214%
	H-FH **	80.82%	**90.5%!**	9.683%
	FT-FH **	86.03%	**90.5%!**	4.469%
o-German	T-H **	69.1%	**72.53%!**	3.434%
	T-FT **	69.1%	**74.62%!**	5.528%
	T-FH **	69.1%	**77.89%!**	8.794%
	H-FT *	72.53%	74.62%!	2.094%
	H-FH **	72.53%	**77.89%!**	5.36%
	FT-FH **	74.62%	**77.89%!**	3.266%
English	T-H **	70.12%	**77.57%!**	7.446%
	T-FT **	70.12%	**77.98%!**	7.858%
	T-FH **	70.12%	**79.86%!**	9.731%
	H-FT =	77.57%	77.98%!	0.4122%
	H-FH **	77.57%	**79.86%!**	2.285%
	FT-FH **	77.98%	**79.86%!**	1.872%
o-English	T-H =	65.77%	66.68%!	0.9091%
	T-FT **	65.77%	**69.83%!**	4.056%
	T-FH **	65.77%	**69.69%!**	3.916%
	H-FT **	66.68%	**69.83%!**	3.147%
	H-FH **	66.68%	**69.69%!**	3.007%
	FT-FH =	69.83%!	69.69%	-0.1399%
Catalan	T-H **	65.23%	**74.76%!**	9.534%
	T-FT **	65.23%	**76.77%!**	11.55%
	T-FH **	65.23%	**78.51%!**	13.28%
	H-FT **	74.76%	**76.77%!**	2.013%
	H-FH **	74.76%	**78.51%!**	3.751%
	FT-FH **	76.77%	**78.51%!**	1.738%
Spanish	T-H **	64.89%	**73.95%!**	9.066%
	T-FT **	64.89%	**75.04%!**	10.15%
	T-FH **	64.89%	**77.61%!**	12.72%
	H-FT *	73.95%	75.04%!	1.082%
	H-FH **	73.95%	**77.61%!**	3.653%
	FT-FH **	75.04%	**77.61%!**	2.571%
Czech	T-H **	65.84%	**73.28%!**	7.437%
	T-FT **	65.84%	**69.99%!**	4.148%
	T-FH **	65.84%	**72.74%!**	6.891%
	H-FT **	**73.28%!**	69.99%	-3.289%
	H-FH **	**73.28%!**	72.74%	-0.5456%
	FT-FH **	69.99%	**72.74%!**	2.743%
o-Czech	T-H **	64.73%	**72.56%!**	7.83%
	T-FT **	64.73%	**69.24%!**	4.502%
	T-FH **	64.73%	**72.23%!**	7.491%
	H-FT **	**72.56%!**	69.24%	-3.328%
	H-FH =	72.56%!	72.23%	-0.3385%
	FT-FH **	69.24%	**72.23%!**	2.989%
Japanese	T-H **	56.69%	**71.92%!**	15.23%
	T-FT **	56.69%	**60.43%!**	3.743%
	T-FH **	56.69%	**70.53%!**	13.84%
	H-FT **	**71.92%!**	60.43%	-11.49%
	H-FH **	**71.92%!**	70.53%	-1.392%
	FT-FH **	60.43%	**70.53%!**	10.1%

	accuracy	1st>2nd	2nd>1st	difference
total	T-H	*0, **0	*9, **9	7.438%
	T-FT	*0, **0	*10, **10	7.109%
	T-FH	*1, **1	*9, **9	9.771%
	H-FT	*3, **3	*6, **4	-0.329%
	H-FH	*3, **3	*6, **6	2.332%
	FT-FH	*1, **1	*8, **8	2.661%



The four bars on Figure 5.3 are drawn linearly.

Figure 5.4: Accuracy of Tree edit distance

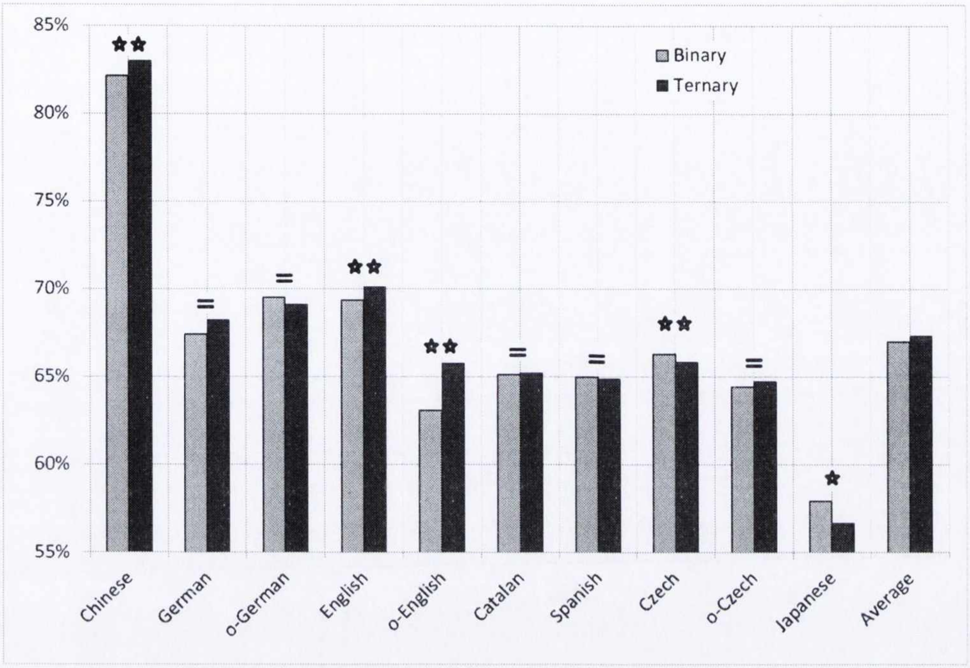
mark at the right side of a percentage in the first or second column indicates which of the values is higher. If the McNemar test detected a strict significant difference then the higher value, in addition to having an exclamation mark, will be written in bold letters. Each language contains six comparisons because they correspond to all the possible comparisons between the four settings.

The bottom of table 5.1 summarises the outcomes of pair-wise comparisons of the swap settings. Under '1st > 2nd', appears two values at the side of on and two asterisks ($*m$,$**n$), the value of (m) represents the amount of significantly different experiments ($p = 0.05$) in which ("1st>2nd"), and n represents the same but with stricter significance level ($p = 0.001$).

The upper side of the table shows the absolute values of accuracy and differences per languages.

This table shows that the hypothesis about the hierarchy of atomic measures is more or less accomplish (see Section 4.7.1): $Shape < Binary < Ternary < Hamming$, $Ternary < Frame\ Ternary$ and $Hamming < Frame\ Hamming$.

5.3.1.1 Binary and Ternary



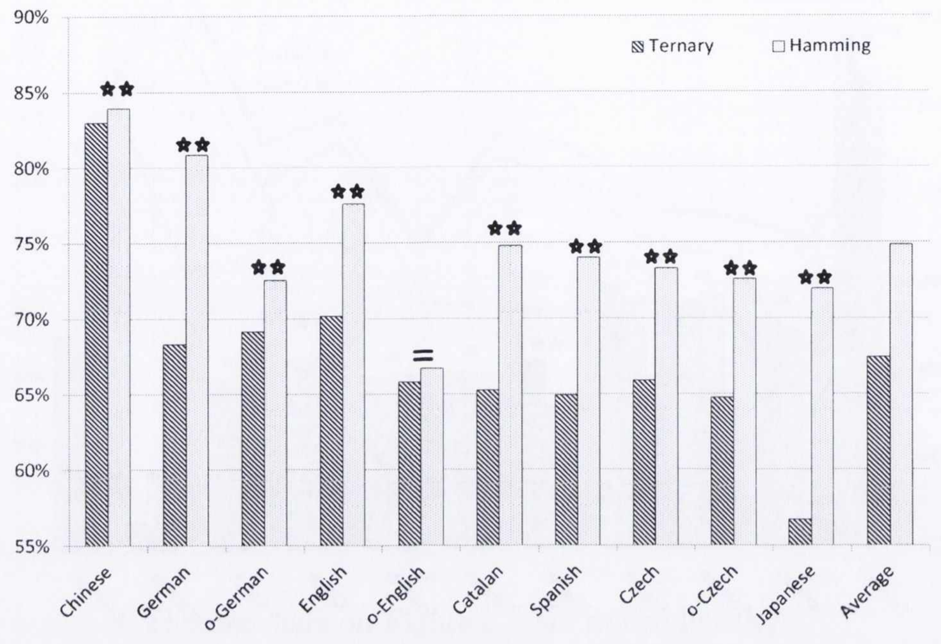
accuracy		Binary	Ternary	difference
Chinese	B-T **	82.14%	**83%!**	0.8624%
German	B-T =	67.41%	68.25%!	0.838%
o-German	B-T =	69.51%!	69.1%	-0.4188%
English	B-T **	69.34%	**70.12%!**	0.7816%
o-English	B-T **	63.08%	**65.77%!**	2.692%
Catalan	B-T =	65.12%	65.23%!	0.1064%
Spanish	B-T =	64.97%!	64.89%	-0.08457%
Czech	B-T **	**66.28%!**	65.84%	-0.436%
o-Czech	B-T =	64.42%	64.73%!	0.3097%
Japanese	B-T *	57.93%!	56.69%	-1.241%
total	B-T	*2, **1	*3, **3	0.341%

Two stars indicate strictly statistical difference, and one start indicates statistical difference.

Figure 5.5: Accuracy of Tree edit distance, Binary and Ternary

Binary performs surprisingly similar to Ternary in spite of its simplicity. Figure 5.5 contrasts both performances and shows the significant differences B-T. This comparison was omitted in Table 5.1 because Binary is considered a base line. In six data sets Ternary performs better than Binary but only three with strictly significant difference. Binary was found to be strictly significantly better in Czech, and significantly better in Japanese. The average difference is as small as 0.341% in favour of Ternary.

5.3.1.2 Ternary and Hamming



accuracy		Ternary	Hamming	difference
Chinese	T-H **	83%	**83.93%!**	0.9238%
German	T-H **	68.25%	**80.82%!**	**12.57%**
o-German	T-H **	69.1%	**72.53%!**	3.434%
English	T-H **	70.12%	**77.57%!**	7.446%
o-English	T-H =	65.77%	66.68%!	0.9091%
Catalan	T-H **	65.23%	**74.76%!**	**9.534%**
Spanish	T-H **	64.89%	**73.95%!**	**9.066%**
Czech	T-H **	65.84%	**73.28%!**	7.437%
o-Czech	T-H **	64.73%	**72.56%!**	7.83%
Japanese	T-H **	56.69%	**71.92%!**	**15.23%**
total	T-H	*0, **0	*9, **9	7.438%

Two stars indicate strictly statistical difference, and one start indicates statistical difference.
The biggest gains are written in bold.

Figure 5.6: Accuracy of Tree edit distance, Ternary and Hamming

Figure 5.6 contrasts Ternary and Hamming performances and shows the significant differences T-H. The accuracy for Ternary ranges from 56% to 83% and it is always out-performed by Hamming with strict significant difference except for the English out-of-domain data set. The average difference between Ternary and Hamming is 7.4%.

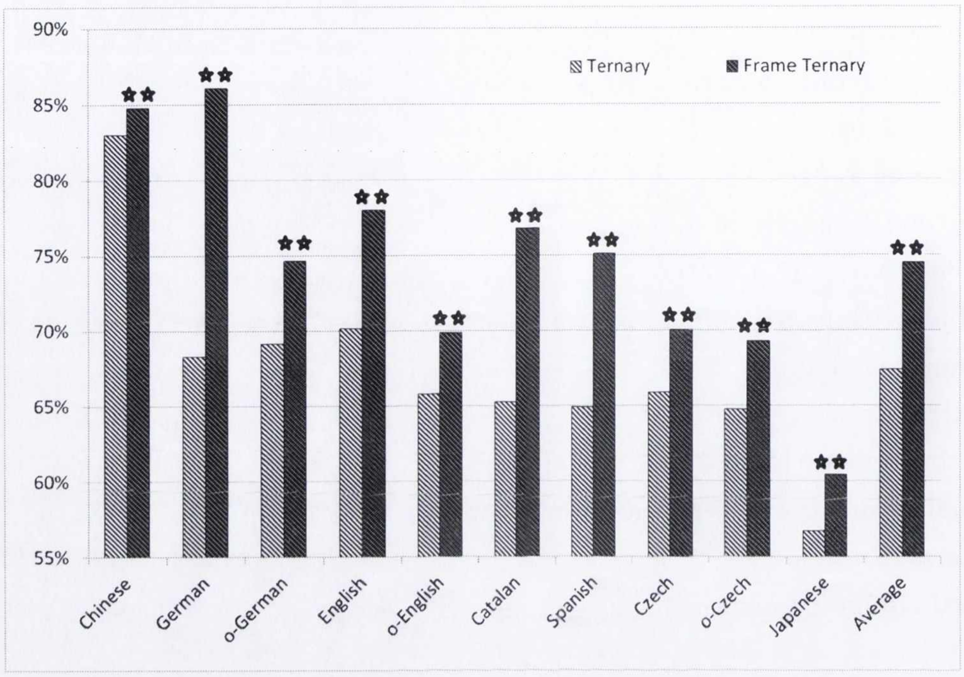
The main difference in design between Ternary and Hamming is that Ternary only uses syntactic features (Dependency Relation and POS) whereas Hamming also uses lexical features (Form and Lemma). The way Hamming settings uses lexical features is very rudimentary: it only checks if the feature has an identical value or not, in general using lexical features increases accuracy but this rudimentary can reduce accuracy in some cases such Chinese, which will be discus in later sections.

Hamming ranges from 66% to 84%. The maximum differences between Ternary and Hamming were found in Japanese (15%) and German (13%).

By reviewing the literature it was observed that it is common to find sophisticated algorithms to evaluate how similar two different words are; for instance Furstenau and Lapata (2011) used a co-occurrence algorithm created from a lemmatized version of the BNC. The system could work substantially better if one of these algorithms were to be used, but just by evaluating if lexical labels are equal or not the system increases its accuracy: Hamming performs better than Ternary.

Word similarity algorithms are left from the scope of this thesis as future work.

5.3.1.3 Ternary and Frame Ternary



accuracy		Ternary	Frame Ternary	difference
Chinese	T-FT **	83%	**84.78%!**	1.779%
German	T-FT **	68.25%	**86.03%!**	17.78%
o-German	T-FT **	69.1%	**74.62%!**	5.528%
English	T-FT **	70.12%	**77.98%!**	7.858%
o-English	T-FT **	65.77%	**69.83%!**	4.056%
Catalan	T-FT **	65.23%	**76.77%!**	11.55%
Spanish	T-FT **	64.89%	**75.04%!**	10.15%
Czech	T-FT **	65.84%	**69.99%!**	4.148%
o-Czech	T-FT **	64.73%	**69.24%!**	4.502%
Japanese	T-FT **	56.69%	**60.43%!**	3.743%
total	T-FT	*0, **0	*10, **10	7.109%

This graph simplifies Figure 5.3.

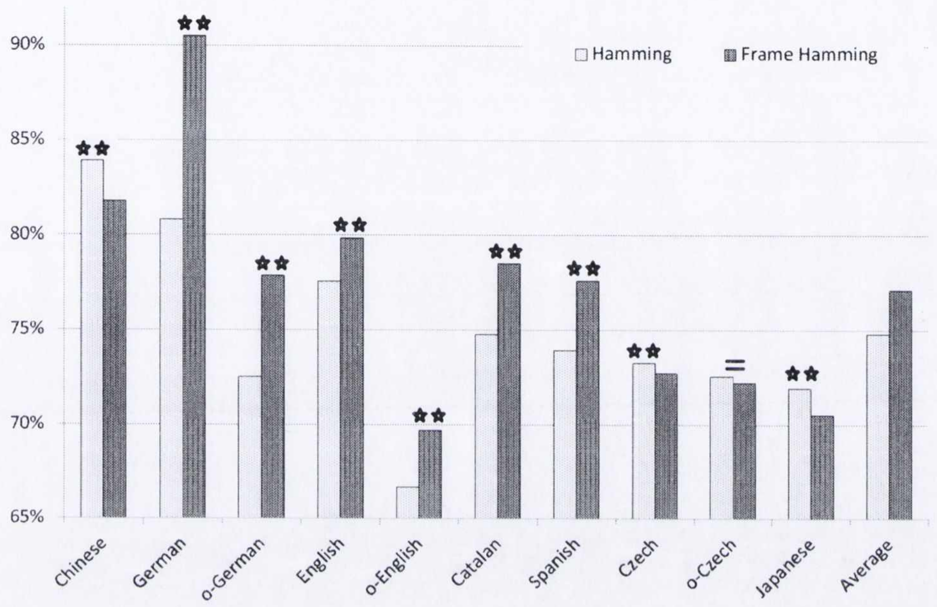
Figure 5.7: Accuracy Ternary vs Frame Ternary

Figure 5.7 contrast Ternary and Frame Ternary. As it was expected,[4] Frame Ternary performs better than Ternary with 7.16% difference on average, it is remarkable that Frame Ternary performs better with strict significant threshold difference for all ten data sets. German, Catalan and Spanish show the higher differences of 17.78%, 11.55% and 10.15%.

[4]As the complexity of the atomic measures increases it is expected the accuracy will increase as well. See section 4.7.1 on page 72 for further details.

Section 5.3.3 will explain further details on about the effect of adding a frame mismatch, on Ternary, Hamming and Shape measures.

5.3.1.4 Hamming and Frame Hamming



accuracy		Hamming	Frame Hamming	difference
Chinese	T-FT **	**83.93%!**	81.79%	-2.14%
German	T-FT **	80.82%	**90.5%!**	9.683%
o-German	T-FT **	72.53%	**77.89%!**	5.36%
English	T-FT **	77.57%	**79.86%!**	2.285%
o-English	T-FT **	66.68%	**69.69%!**	3.007%
Catalan	T-FT **	74.76%	**78.51%!**	3.751%
Spanish	T-FT **	73.95%	**77.61%!**	3.653%
Czech	T-FT **	**73.28%!**	72.74%	-0.5456%
o-Czech	T-FT =	72.56%!	72.23%	-0.3385%
Japanese	T-FT **	**71.92%!**	70.53%	-1.392%
total	T-FT	*3, **3	*6, **6	2.332%

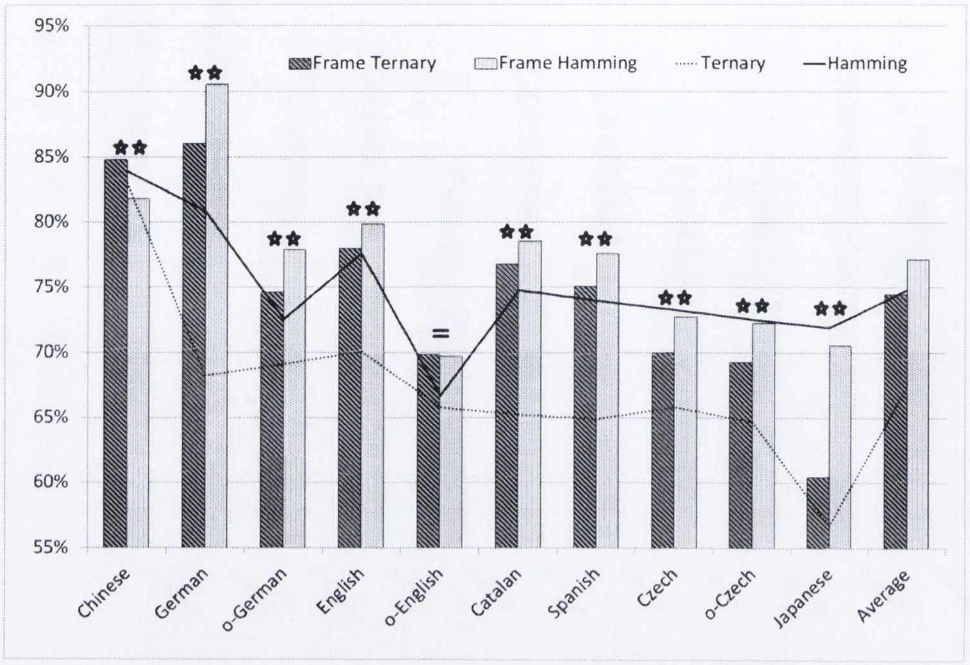
This graph simplifies Figure 5.3.

Figure 5.8: Accuracy Hamming vs Frame Hamming

Figure 5.8 illustrates the comparison between Hamming and Frame Hamming settings.

As it was expected Frame Hamming performs better than Hamming, but not as notable as Ternary vs Frame Ternary. In three of ten data sets: Chinese, Czech and Japanese, Hamming performs better than Frame Hamming with an strictly significant threshold, and in out-of-domain Czech, Hamming also performs better than Frame Hamming but no significant difference was found.

5.3.1.5 Frame Ternary and Frame Hamming

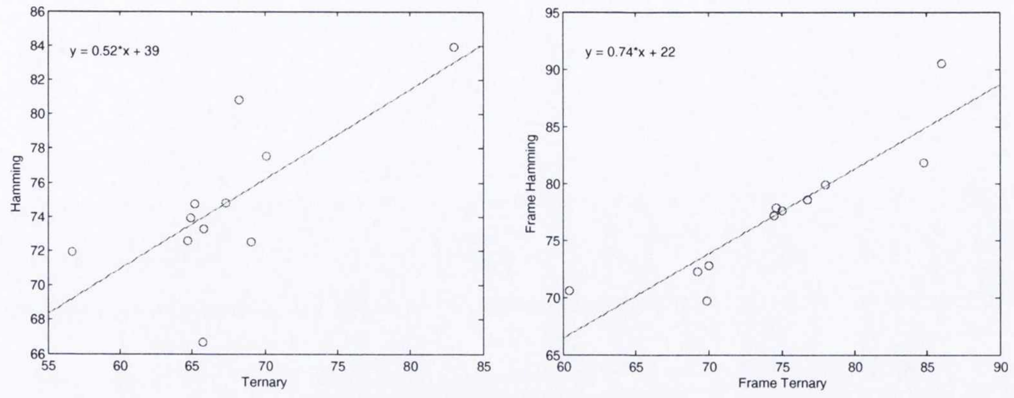


accuracy		Frame Ternary	Frame Binary	difference
Chinese	FT-FH **	**84.78%!**	81.79%	-2.995%
German	FT-FH **	86.03%	**90.5%!**	4.469%
o-German	FT-FH **	74.62%	**77.89%!**	3.266%
English	FT-FH **	77.98%	**79.86%!**	1.872%
o-English	FT-FH =	**69.83%!**	69.69%	-0.1399%
Catalan	FT-FH **	76.77%	**78.51%!**	1.738%
Spanish	FT-FH **	75.04%	**77.61%!**	2.571%
Czech	FT-FH **	69.99%	**72.74%!**	2.743%
o-Czech	FT-FH **	69.24%	**72.23%!**	2.989%
Japanese	FT-FH **	60.43%	**70.53%!**	10.1%
total	FT-FH	*1, **1	*8, **8	2.661%

Figure 5.9: Contrasting Frame Ternary and Frame Hamming

Figure 5.9 shows the accuracy of Frame Ternary and Frame Hamming in Bars and Ternary and Hamming in lines, so differences between both can be seen easily.

Frame Ternary ranges from 60% to 86% and Frame Hamming from 69% to 91%. Frame Ternary and Frame Hamming seems to track each other more



(a) Ternary and Hamming correlation, Norm of residuals 10.328

(b) Frame Ternary and Frame Hamming correlation, Norm of residuals 8.0734

Figure 5.10: Accuracy correlation between Ternary to Hamming and Frame Ternary to Frame Hamming and their linear regressions

than Ternary and Hamming, as it can be observed on Figure 5.10 the graph correlating Frame Hamming with Frame Ternary have a Norm of residuals of about 8 which is lower than the norm of residuals of the correlation between Ternary and Hamming which is 10.3. Both norm of residuals are relatively high.

The differences between Frame Ternary and Frame Hamming (average difference is 2.7%) are smaller than the differences between Ternary and Hamming (average difference is 7.4%).

Three data sets seem to be different. Chinese, because Frame Hamming has lower accuracy than Hamming, o-English because Frame Ternary and Frame Hamming seems to have the same accuracy; and Japanese because the differences are the largest among all data sets.

The Chinese case is due to the particularly low accuracy of the Frame Hamming setting, which performs worse than any of the other three measures. This will be analysed in Section 5.3.2.3.

Figure 5.10 shows the correlations, Figure 5.10a shows the correlation between Ternary and Hamming and Figure 5.10b shows the correlation between Frame Ternary and Frame Hamming. Frame Ternary and Frame Hamming show better correlation with each other (Norm residuals = 8.0734) than Ternary and Hamming (Norm of residuals = 10.328).

The Norm of Residuals ($|z|$)[5] is calculated as:

(5.1) $$|z| = \sqrt{\sum_{l} Acc_l^2 - f(Acc_l^1)}$$

Where l is each data set used, Acc_l^1 is the accuracy for the first setting (Ternary or Frame Ternary) on the language l, Acc_l^2 is the accuracy for the second setting (Hamming or Frame Hamming) on the language l and $f(x)$ is the value y for the linear regression at position x.

5.3.1.6 Chinese

Chinese has the highest overall accuracy across languages, with even the very simplest settings reaching relatively high accuracy. Section 5.3.6 will show more observations of this phenomenon along with the German data.

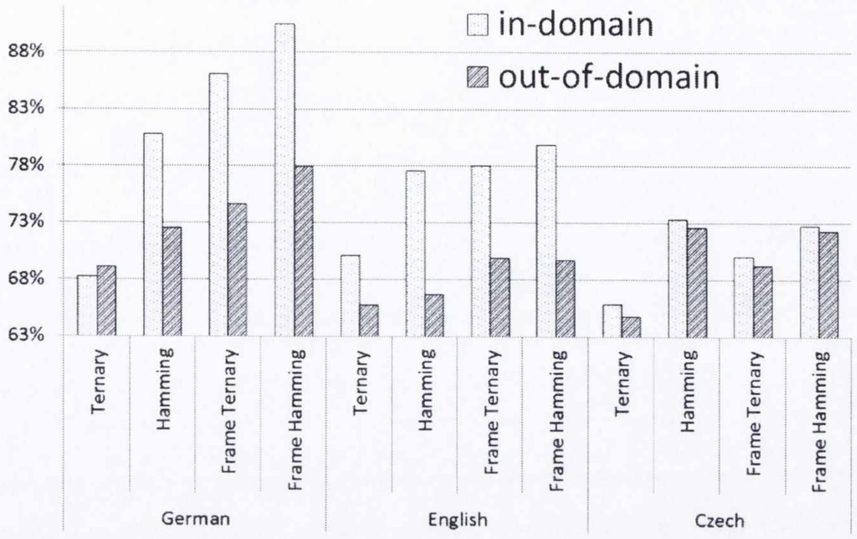
5.3.1.7 Japanese

Japanese reports the worst in-domain results, especially for the two Ternary versions settings. This is probably due to the fact that, on inspection, the Japanese data gives 96.1% of syntactic dependencies the same dependency relation,[6] practically cancelling the contribution of the dependency relation feature.

An alternative, less satisfactory explanation is that the average amount of nodes per sub-tree is very high, which makes it difficult to find an adequate mapping. Figure 5.13 in Section 5.3.8 illustrates this phenomenon.

5.3.1.8 Spanish and Catalan

The outcomes for the Spanish and Catalan data sets are very similar to each other. This is not unexpected as for the most part one is a translation of the other and they were annotated in the same way with the same set of labels. This was already mention at Section 2.4.2 on page 22.



		in-domain	out-of-domain	fall
German	Ternary	68.25%	69.10%	**-0.85%**
	Hamming	80.82%	72.53%	8.29%
	Frame Ternary	86.03%	74.62%	11.41%
	Frame Hamming	90.50%	77.89%	12.61%
English	Ternary	70.12%	65.77%	**4.35%**
	Hamming	77.57%	66.68%	10.89%
	Frame Ternary	77.98%	69.83%	8.15%
	Frame Hamming	79.86%	69.69%	10.17%
Czech	Ternary	65.84%	64.73%	**1.11%**
	Hamming	73.28%	72.56%	**0.72%**
	Frame Ternary	69.99%	69.24%	**0.75%**
	Frame Hamming	72.74%	72.23%	**0.51%**

Figure 5.11: Out-of-domain falls

5.3.1.9 Out-of-domain

Regarding the out-of-domain evaluations (see Figure 5.11), the accuracy of the settings get degraded ~10% in both hamming measures and slightly less in the Ternary measures. This was expected because Ternary measures are completely independent of the lexical features, which makes them more independent of the domain. It is remarkable that all measures in Czech and the Ternary measure in German fail to drop from in-domain to out-of-domain as much as might be expected (drops are around one per cent). In German it even seems to increase the accuracy ~2% in Ternary a bit. One of the reasons for the German case could

[5]This definition of Norm of residuals happens to be the default measure of the graphical tool in which the graph 5.10 was made (Matlab)

[6]It gives a perplexity for all dependency relations used of 1.17

be that the accuracy of the in-domain data set was already very low relative to other measurements. Hajic et al. (2009) mentioned that the German out-of-domain data set was sampled from the EUROPARL corpus (Koehn, 2005) and was chosen to maximize the lexical coverage containing a large number of infrequent predicates. This could explain that the Ternary setting, which does not use lexical features, was not affected by the change of domain, but all other settings were affected.

It is difficult to explain why Czech is relatively unaffected by a change of the evaluation domain. A clue to finding this could be that the order of the arguments in the out-of-domain data set is more similar to the training data set than the in-domain evaluation data. Table 5.2, shows the perplexity produced by the sequence of artificial labels created by concatenating all semantic argument labels for each predicate, plus the predicate, according to their word order. For instance, if a predicate "sell.01" has two arguments, the first one is "A0" and the second one is "A1" the representative label for that predicate will be "A0A1sell.01". As can be seen, the perplexity of Czech for the training data set plus the out-of-domain evaluation data set is lower than the training data set plus the in-domain evaluation data set. It suggests that the order of arguments of the out-of-domain data set is more similar to the training data set than the in-domain evaluation data set.

Another clue may be that the perplexity of frames of Czech is higher than that of any other language, suggesting that the training data set has a very broad topic scope and it may include the out-of-domain as a sub-topic. See Figure 5.12 at page 94.

Table 5.2: Perplexity of argument sequences plus predicate in and out-of-domain

Perplexity	training +evaluation	training +odd	difference	ternary fall	hamming fall
German	940.47	973.80	33.33	-0.85	8.29
English	13,160.40	13,164.60	4.20	4.35	10.89
Czech	18,442.30	18,271.10	**-171.20**	1.11	0.72

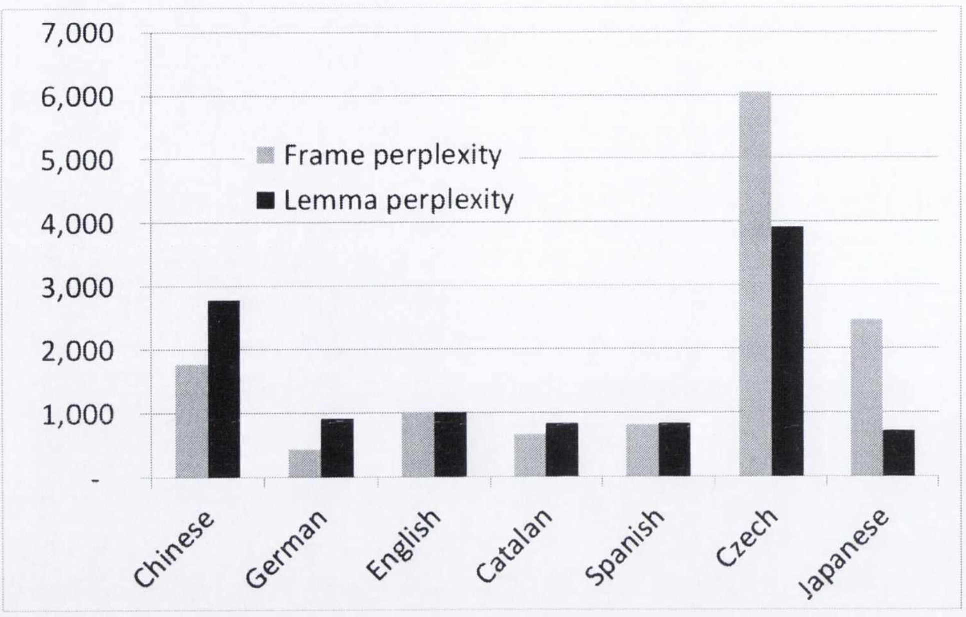
Absolute values and differences are not comparable across languages because the size of the in-domain evaluation and out-of-domain evaluation data sets are different, and the set of semantic roles are different as well. The important observation is if the difference between perplexity in and out-of-domain is positive or negative.

5.3.2 Frame Hamming < Hamming

In four out of ten cases, Frame Hamming scores lower than Hamming (See Table 5.8 at page 87): for Chinese, Czech and Japanese it happens with a strict significant difference and for out-of-domain Czech it happens without a significant significant difference.

One side effect of adding cost for frame mismatches can be a larger demand of samples per frame, otherwise bad samples from a same frame may be promoted. Hamming may be more susceptible to such effect than Ternary, because lexical features are very diverse in contrast with syntactic features. Therefore, Hamming is more likely to promote samples which have more common words, which may not be good samples.

Another way to see it, could be that Chinese, Czech and Japanese have too many frames, perhaps the frame set is too detailed or each frame can have too many senses/meanings. Figure 5.12 show that these three languages have the higher frame perplexity of all data sets used.



Language	Frame perplexity	Lemma perplexity	Amount sub-trees	samples per frame perplexity
Chinese	1,774	2,787	102,813	57.96
German	431	911	17,400	40.36
English	1,009	1,009	91,250	90.45
Catalan	662	840	37,431	56.56
Spanish	801	840	43,824	54.69
Czech	6,054	3,919	106,138	**17.53**
Japanese	2,465	719	25,712	**10.43**

Figure 5.12: Perplexity of frames and lemmas
Training data sets only

5.3.2.1 Out-of-domain Czech

In out-of-domain Czech, the differences between Hamming and Frame Hamming are not significant. One might expect that Frame Hamming is more likely to score lower than Hamming in out-of-domain data than in-domain data because the out-of-domain data is more likely to contain infrequent frames. As Frame Hamming will find very few samples of the same frame. Hence it will damage the performance of Frame Hamming more than Hamming.

5.3.2.2 Czech and Japanese

Czech and Japanese languages seem to have very high Frame perplexity for the amount of samples in the data set. Its perplexity suggests an average of 17.53 samples per frame for Czech and 10.43 samples per frame for Japanese as seen in Figure 5.12, where the other data sets have between 40 to 90. It could be that those amounts of samples are insufficient for Frame Hamming to work properly.

This hypothesis is re-enforced by a experiment run with the full data set of Czech. Table 5.3 shows that once the whole Czech data set is used to train the machine, the differences between Hamming and Frame Hamming become smaller and with the full data set there was no significant difference found.

Table 5.3: Czech Hamming vs Frame Hamming

accuracy		Hamming	Frame Hamming	difference
reduce data set	H-FH **	**73.28%!**	72.74%	**-0.5456%**
full data set	H-FH =	77.11%!	77.08%	**-0.03569%**

5.3.2.3 Chinese Frame Hamming

The under-performance of Frame Hamming on the Chinese data set is probably the most surprising of all, because it performs worse than the other three measures used.

A possible explanation about why the accuracy for Frame Hamming in the Chinese data set is particularly low could be that the variety of the lexical features with the variability of the predicate frames was especially high.

As Figure 5.12 shows, the Chinese data set has very high lexical perplexity and very high frame perplexity. Only the Czech data set has higher lexical perplexity than the Chinese.

Having very high perplexity of Lemma labels and very high perplexity of predicate frames makes the lexical features potential noise for the k-NN system, as they may be often too different. Therefore, in this situation it is not possible for the k-NN module to discriminate useful samples from the useless ones. This explains why Frame Hamming has the lowest accuracy of all measures compared (T,H and FT) when for all other languages it usually presents the higher results.

The high frame perplexity seems is due to the annotators as they probably decided to annotate a large amount of frames, it is important to note that Chinese has the third highest frame perplexity across the seven languages.

The high lemma perplexity is probably due to the selection of a broad language domain rather than a intrinsic feature of the Chinese language. The question about if the Chinese language is richer in lemmas is out of the scope of this thesis.

5.3.3 Simple vs Frame

Table 5.4: Simple vs Frame matching

accuracy		frame mismatch+0	frame mismatch+1	difference
Chinese	S-FS **	47.49%	**56.31%!**	8.812%
	T-FT **	83%	**84.78%!**	1.779%
	H-FH **	**83.93%!**	81.79%	-2.14%
German	S-FS **	54.66%	**73.93%!**	19.27%
	T-FT **	68.25%	**86.03%!**	17.78%
	H-FH **	80.82%	**90.5%!**	9.683%
o-German	S-FS *	62.06%	65.83%!	3.769%
	T-FT **	69.1%	**74.62%!**	5.528%
	H-FH **	72.53%	**77.89%!**	5.36%
English	S-FS **	44.24%	**58.05%!**	13.81%
	T-FT **	70.12%	**77.98%!**	7.858%
	H-FH **	77.57%	**79.86%!**	2.285%
o-English	S-FS **	44.27%	**50.17%!**	5.909%
	T-FT **	65.77%	**69.83%!**	4.056%
	H-FH **	66.68%	**69.69%!**	3.007%
Catalan	S-FS **	26.19%	**52.02%!**	25.83%
	T-FT **	65.23%	**76.77%!**	11.55%
	H-FH **	74.76%	**78.51%!**	3.751%
Spanish	S-FS **	26.73%	**50.74%!**	24.01%
	T-FT **	64.89%	**75.04%!**	10.15%
	H-FH **	73.95%	**77.61%!**	3.653%
Czech	S-FS **	35.17%	**50.82%!**	15.66%
	T-FT **	65.84%	**69.99%!**	4.148%
	H-FH **	**73.28%!**	72.74%	-0.5456%
o-Czech	S-FS **	35.18%	**50.89%!**	15.71%
	T-FT **	64.73%	**69.24%!**	4.502%
	H-FH =	72.56%!	72.23%	-0.3385%
Japanese	S-FS **	35.47%	**47.47%!**	12%
	T-FT **	56.69%	**60.43%!**	3.743%
	H-FH **	**71.92%!**	70.53%	-1.392%
total	S-FS	*0, **0	*10, **9	14.48%
	T-FT	*0, **0	*10, **10	7.109%
	H-FH	*3, **3	*6, **6	2.332%
	all	*3, **3	*26 ,**25	7.973%

In order to investigate the effect of adding a cost for frame mismatch, further experiments were done adding Shape vs Frame Shape along with Ternary/Frame Ternary and Hamming/Frame Hamming.

Table 5.4 compares two settings in which the first column does not get any penalty for swapping two nodes of different predicates and the second one gets a cost of one. The atomic measures selected are Shape (baseline), Ternary and Hamming.

As the accuracy of the basic settings is higher, the gain by adding a cost for swapping nodes of different frames decreases, for instance Frame Shape always outperforms Shape with an average difference of 14.48% a significant difference ($p < .05$) was observed ten times and strictly significant difference ($p < .001$) was observed nine times. Frame Ternary performs better than Frame Shape, and it always outperforms Ternary with strictly significant difference and an average difference of 7.1%.

The biggest difference is found in the shape setting, the second biggest difference in Ternary, and the smallest differences are found in Hamming, which on some occasions even reports a small decrement of accuracy.

As said in previous paragraphs the settings with a cost for matching nodes from different frames report higher accuracy, with three exceptions on the Hamming versions, Chinese, Czech and Japanese.

5.3.4 Frame flexible vs extreme

Table 5.5: Flexible frame vs strict

accuracy		frame mismatch=+1	frame mismatch=+100	difference
Chinese	FS **	**56.31%!**	48.66%	-7.65%
	FT **	**84.78%!**	68.21%	-16.58%
	FH **	**81.79%!**	68.45%	-13.34%
German	FS **	**73.93%!**	70.39%	-3.538%
	FT =	86.03%	87.62%!	1.583%
	FH *	90.5%!	88.73%	-1.769%
o-German	FS **	**65.83%!**	44.97%	-20.85%
	FT **	**74.62%!**	53.43%	-21.19%
	FH **	**77.89%!**	54.27%	-23.62%
English	FS **	**58.05%!**	52.69%	-5.368%
	FT **	**77.98%!**	71.36%	-6.626%
	FH **	**79.86%!**	72.76%	-7.098%
o-English	FS **	**50.17%!**	38.46%	-11.71%
	FT **	**69.83%!**	52.34%	-17.48%
	FH **	**69.69%!**	53.6%	-16.08%
Catalan	FS **	**52.02%!**	49.62%	-2.403%
	FT **	**76.77%!**	73.45%	-3.326%
	FH **	**78.51%!**	73.21%	-5.303%
Spanish	FS **	**50.74%!**	48.48%	-2.258%
	FT **	**75.04%!**	71.95%	-3.087%
	FH **	**77.61%!**	72.67%	-4.939%
Czech	FS **	**50.82%!**	39.69%	-11.13%
	FT **	**69.99%!**	52.22%	-17.77%
	FH **	**72.74%!**	53.13%	-19.61%
o-Czech	FS **	**50.89%!**	40.02%	-10.87%
	FT **	**69.24%!**	52.81%	-16.43%
	FH **	**72.23%!**	53.58%	-18.64%
Japanese	FS **	**47.47%!**	33.18%	-14.29%
	FT **	**60.43%!**	41.45%	-18.98%
	FH **	**70.53%!**	46.15%	-24.37%
total	FS	*10, **10	*0, **0	-9.008%
	FT	*9, **9	*0, **0	-11.99%
	FH	*10, **9	*0, **0	-13.48%
	all	*29, **28	*0 ,**0	-11.49%

The previous section shows that adding a cost on swapping nodes of different frame (or predicate nodes to non-predicate nodes) increases accuracy. These results suggest that using samples from different frames may always lead to a decrement in accuracy. If that is the case, it should work even better when the mismatch cost tends to infinite. In order to test this hypothesis, Table 5.5 compares the results of the settings for an extra cost of adding one unit against adding a hundred units if the predicate nodes belong to different frames.

Table 5.5 shows the results in which it can be seen clearly, that contrary to what one might expect, removing the samples of other frames has a negative impact. The only exception is German Ternary in which the accuracy improves without any significant difference.

5.3.4.1 Error analysis

The error analysis for the English in-domain data over Hamming shows that a substantial amount of predictions change from correct to incorrect and vice versa, but the average is negative because ~58% of the changes are to the wrong side.

From the predictions that become incorrect, ~ 38% are due to the system not being able to predict any label. This can be a consequence of not finding a single example that can project a label. Which probably is the result of banning the usage of examples from different frames.

If all the cases where frame extreme (banning use of samples of different frame) were to adopt the output of the first setting (which just adds an extra cost) the performance will be 78.02% which is still less than the first setting. It reinforces the hypothesis that banning samples from different frames decreases accuracy.

5.3.5 Shape

Observing in detail the log files produced by the shape settings it can be observed that occasionally the system is not able to produce a prediction, occurring 5.5% on average for all data sets. This type of failure is very dependent on the language, sometimes it is due to a draw in the voting and sometimes due to not finding any useful mapping. For instance in the case of Japanese 48% of the non-predicted cases were due to not finding any useful mapping. The percentages of failures to predict a label in descending order per data set are: Japanese 19.98%, Czech out-of-domain 10.72%, Czech 9.72%, English out-of-domain 3.92%, Chinese 3.59%, English 3.57%, German out-of-domain 1.68%, German 1.12%, Catalan 0.78%, and Spanish 0.08%. Spanish and Catalan ratios of unpredicted labels are especially low, lower than 1%.

The fact that Spanish and Catalan had a very low ratio of the system failing to make any prediction does not mean the system works well. As can be seen in Figure 5.2 at page 78, Shape produced its worst results for Spanish and Catalan.

Table 5.6: Top three system performance in CoNLL-2009

Che (Che et al., 2009)		Shen (Dai et al., 2009)		Merlo (Gesmundo et al., 2009)	
87.00	English	87.69	English	86.03	English
85.65	Japanese	85.28	Japanese	84.91	Japanese
83.27	Czech	83.31	Spanish	83.21	Czech
82.64	**Average**	83.01	Catalan	82.66	Catalan
82.44	**German**	**82.52**	**Average**	82.43	Spanish
81.90	Spanish	81.22	**German**	**82.14**	**Average**
81.84	Catalan	80.87	Czech	79.59	**German**
76.38	**Chinese**	76.23	**Chinese**	76.15	**Chinese**

These data sets are among the ones with the least amount of nodes per sub-tree, with makes them more likely to have a similar shape along different sub-trees. It also appears that all sub-trees for Spanish and Catalan are rooted on the predicate, which causes the sub-trees more likely to have the same shape and the mappings to be usable. One of the optimizations for the system to save memory was to save only the distance to the first nearest neighbours. This means that if a the first equivalence class has more samples than the limit, then that equivalence class will be truncated, by removing the rest of the samples. Therefore the k-NN module can use only a panel size up to that amount. As the Shape setting does not distinguish on the content of the nodes, the first equivalence class is always at least one thousand samples, so all the distances saved are equal.

5.3.6 The Chinese and German out-performance

Table 5.6 shows the F_1 score for each language of the three best performing algorithms in the CoNLL-2009 evaluation close challenge, sorting the language by its performance. All three settings give the worst results in the Chinese data set, which is interesting because the tree-SRL system gives the best average results on that particular data set. Again, for all three settings the results of the German data are below the average which is also interesting because in Tree-SRL the results on German data are second best on average and achieve the highest performance of all experiments for Frame Hamming settings. It suggests that tree distance and labelling by alignment methods may be especially suitable for Chinese and German data sets. Another surprise is Japanese, which gives the worst results in the Tree-SRL but the second best for Chen, Shen and Merlo systems.

5.3.7 German data set

Table 5.7: Accuracy for all labels and for A0..A9 together

accuracy			simple	frame	difference
Chinese	all	T-FT **	83%	**84.78%!**	1.779%
	all	H-FH **	**83.93%!**	81.79%	-2.14%
	A0-9	T-FT **	87.06%	**90.44%!**	**3.38%**
	A0-9	H-FH **	88.7%	**90.03%!**	**1.339%**
German	all	T-FT **	68.25%	**86.03%!**	17.78%
	all	H-FH **	80.82%	**90.5%!**	9.683%
	A0-9	T-FT **	68.25%	**86.03%!**	17.78%
	A0-9	H-FH **	80.82%	**90.5%!**	9.683%
o-German	all	T-FT **	69.1%	**74.62%!**	5.528%
	all	H-FH **	72.53%	**77.89%!**	5.36%
	A0-9	T-FT **	69.1%	**74.62%!**	5.528%
	A0-9	H-FH **	72.53%	**77.89%!**	5.36%
English	all	T-FT **	70.12%	**77.98%!**	7.858%
	all	H-FH **	77.57%	**79.86%!**	2.285%
	A0-9	T-FT **	73.78%	**83.3%!**	**9.521%**
	A0-9	H-FH **	82.08%	**86.63%!**	**4.55%**
o-English	all	T-FT **	65.77%	**69.83%!**	4.056%
	all	H-FH **	66.68%	**69.69%!**	3.007%
	A0-9	T-FT **	73.16%	**78.95%!**	**5.795%**
	A0-9	H-FH **	74.79%	**80.44%!**	**5.646%**
total	all	T-FT	*0, **0	*5, **5	7.401%
	all	H-FH	*1, **1	*4, **4	3.639%
	all	all	*1, **1	*9 ,**9	5.52%
	A0-9	T-FT	*0, **0	*5, **5	**8.402%**
	A0-9	H-FH	*0, **0	*5, **5	**5.316%**
	A0-9	all	*0, **0	*10 ,**10	**6.859%**

The German data set extracted from SALSA (Burchardt et al., 2006) was originally annotated in FrameNet style and then semi-automatically translated to Propbank style. In the process the annotation was substantially simplified, which explains why the German data set has the lowest semantic labels perplexity of all data sets (probably the data were over-simplified).

It seems that adjuncts are not annotated, for example the word "yesterday" is usually annotated in the English data set as an "AM-TMP" role of a predicate, but the word "gestern" which means yesterday in German, is not annotated as a semantic role.

The high performance of the Frame version could be a side effect produced by a good performance of the core arguments, which are more predicate-dependent and the fact that the adaptation for the CoNL2009 substantially reduce the annotation of adjuncts.

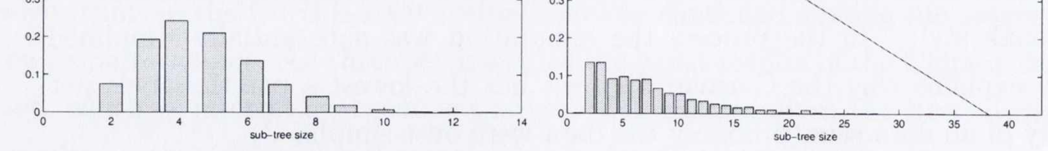
In that case it would be expected that for all languages the Frame versions will out-perform the simpler ones in core arguments. Table 5.7 shows the accuracy of the simple Ternary and Hamming settings against their frame versions for all arguments, and for the enumerated arguments "A0" to "A9", which are shared among German, English and Chinese[7].

As it was expected, the differences between the frame version and simple version become larger when only core arguments are evaluated. It suggests that giving priority for samples of the same frame helps core arguments more than adjuncts.

German language does not show any difference because it only has enumerated arguments. Therefore the results have to be identical.

5.3.8 Number of nodes per sub-tree

It seems to be a clear inverse correlation along all data sets (all languages) between accuracy and the number of nodes on the sub-tree to be labelled.



(a) Chinese (b) Japanese

Figure 5.13: Ternary accuracy along number of nodes per sub-tree

[7]the other languages do not use "A0" to "A9" labels.

Figures 5.13a and 5.13b for Chinese and Japanese ternary respectively.

The x-axis represents the amounts of nodes per sub-tree. Bars represent the percentage of samples at each amount of nodes (the total equals one). The linear line is the linear regression of the accuracy, and the dotted line is the cubic regression.

The x-values in the Japanese graph are larger because there are Japanese sub-trees with a larger amount of nodes.

5.4 Contrasting representations:

5.4.1 Tree vs Linear (Levenshtein)

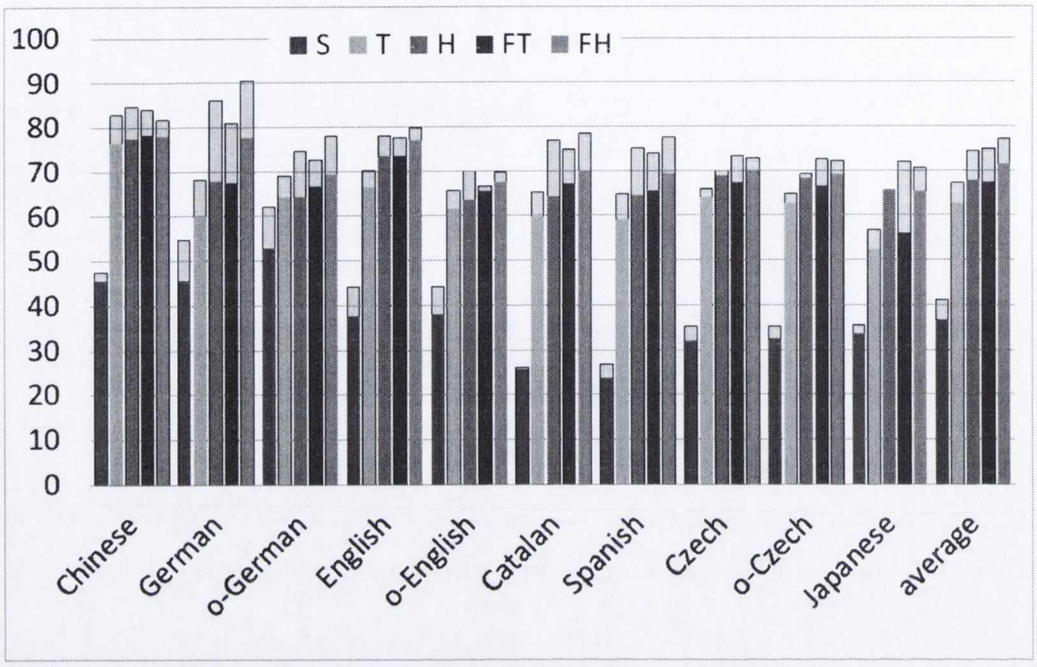
Structural information is important. That is why this thesis examines tree algorithms and not just string algorithms. In order to observe the effect of structural information more clearly a new experiment was designed in which sentences were not parsed and Levenshtein (1966) distance was used in the calculations.

There is no Levenshtein implementation but a particular way to load trees into memory, in which each node still contains the dependency relation to its own head node. However, the sentences trees are represented as a string, one child per node, in the same order as was written in the sentence: the first word being the head of the tree and the last word of the sentence the unique leaf.

In order to observe the effects of removing the structural information in an isolated way, it is important to include the information about the dependency relation, even if there is no specification for which is the head word node. Otherwise the effect may be produced by removing the dependency relations rather than produced by removing the tree structures.

The way to extract samples or sub-trees is exactly the same without any change of the algorithm; the only difference is that the extracted sub-trees are strings.

In many different computational problems excessive information leads machine learning to degrade accuracy, and that is why feature reduction algorithms exist. Thus removing structural information could improve accuracy. However, the results imply clearly that structural information has a very important role in the functionality of the system.



The top value corresponds to tree edit distance, the border value corresponds to linear distance. Both with deletion cost equal to one.

Figure 5.14: Drop of accuracy from tree distance to linear distance

Table 5.8 compares the results obtained with trees to results obtained with the linearised version.

Figure 5.14 shows the comparison graphically, where the solid bars correspond to the values for linear trees, and the top of the bars represent the accuracy for tree edit distance.

As is evident, for all languages and almost all swap-settings, the alignment on the linear representation gives substantially poorer results than the alignment on the trees with an exception for Japanese Hamming. Almost all experiments imply that the string versions produce losses on accuracy with a strictly significant significance. The average loss is approximately 5.78%, but not all losses are equal. The frame versions are the most affected by the loss of structural information.

Table 5.8: Tree distance vs linear

accuracy		tree distance	linear distance	difference
Chinese	S **	**47.49%!**	45.67%	-1.826%
	T **	**83%!**	76.59%	-6.416%
	H **	**84.78%!**	77.33%	-7.451%
	FT **	**83.93%!**	78.21%	-5.716%
	FH **	**81.79%!**	78.11%	-3.677%
German	S **	**54.66%!**	45.62%	-9.032%
	T **	**68.25%!**	60.06%	-8.194%
	H **	**86.03%!**	67.78%	-18.25%
	FT **	**80.82%!**	67.5%	-13.31%
	FH **	**90.5%!**	77.56%	-12.94%
o-German	S **	**62.06%!**	52.93%	-9.129%
	T **	**69.1%!**	64.24%	-4.858%
	H **	**74.62%!**	64.24%	-10.39%
	FT **	**72.53%!**	66.75%	-5.779%
	FH **	**77.89%!**	69.43%	-8.459%
English	S **	**44.24%!**	37.56%	-6.678%
	T **	**70.12%!**	66.41%	-3.715%
	H **	**77.98%!**	73.3%	-4.681%
	FT **	**77.57%!**	73.38%	-4.187%
	FH **	**79.86%!**	77.01%	-2.847%
o-English	S **	**44.27%!**	38.08%	-6.189%
	T **	**65.77%!**	61.5%	-4.266%
	H **	**69.83%!**	63.64%	-6.189%
	FT =	66.68%!	65.63%	-1.049%
	FH *	69.69%!	67.48%	-2.203%
Catalan	S =	26.19%!	25.83%	-0.3547%
	T **	**65.23%!**	59.85%	-5.374%
	H **	**76.77%!**	64.31%	-12.46%
	FT **	**74.76%!**	67.06%	-7.698%
	FH **	**78.51%!**	69.81%	-8.7%
Spanish	S **	**26.73%!**	23.79%	-2.943%
	T **	**64.89%!**	59.12%	-5.767%
	H **	**75.04%!**	64.47%	-10.56%
	FT **	**73.95%!**	65.55%	-8.406%
	FH **	**77.61%!**	69.39%	-8.22%
Czech	S **	**35.17%!**	32.07%	-3.1%
	T **	**65.84%!**	64.02%	-1.825%
	H **	**69.99%!**	68.94%	-1.055%
	FT **	**73.28%!**	67.4%	-5.882%
	FH **	**72.74%!**	69.87%	-2.868%
o-Czech	S **	**35.18%!**	32.47%	-2.708%
	T **	**64.73%!**	62.68%	-2.053%
	H *	69.24%!	68.33%	-0.9076%
	FT **	**72.56%!**	66.47%	-6.094%
	FH **	**72.23%!**	69.26%	-2.96%
Japanese	S *	35.47%!	33.61%	-1.862%
	T **	**56.69%!**	52.32%	-4.363%
	**H ** **	60.43%	**65.85%!**	5.417%
	FT **	**71.92%!**	56.03%	-15.89%
	FH **	**70.53%!**	65.41%	-5.116%
total	S	*9, **8	*0, **0	-4.382%
	T	*10, **10	*0, **0	-4.683%
	H	*9, **8	*1, **1	-6.652%
	FT	*9, **9	*0, **0	-7.402%
	FH	*10, **9	*0, **0	-5.799%
	all	*47, **44	*1 ,**1	-5.784%

Table 5.9: Tree vs linear without dependency relations

accuracy		tree	linear	difference
Chinese	S **	**47.49%!**	45.67%	-1.826%
	T **	**73.99%!**	68.53%	-5.452%
	H **	**78.36%!**	70.93%	-7.433%
	FT **	**76.34%!**	70.37%	-5.972%
	FH **	**75.53%!**	71.21%	-4.319%
German	S **	**54.66%!**	45.62%	-9.032%
	T **	**64.34%!**	54.47%	-9.87%
	H **	**80.07%!**	64.15%	-15.92%
	FT **	**84.45%!**	63.59%	-20.86%
	FH **	**86.69%!**	73.09%	-13.59%
o-German	S **	**62.06%!**	52.93%	-9.129%
	T **	**68.43%!**	58.79%	-9.631%
	H **	**72.45%!**	60.55%	-11.89%
	FT **	**73.45%!**	62.48%	-10.97%
	FH **	**76.47%!**	66.25%	-10.22%
English	S **	**44.24%!**	37.56%	-6.678%
	T **	**62.7%!**	60.02%	-2.675%
	H **	**74.91%!**	69.37%	-5.54%
	FT **	**72.88%!**	68.33%	-4.543%
	FH **	**76.84%!**	73.26%	-3.573%
o-English	S **	**44.27%!**	38.08%	-6.189%
	T **	**56.64%!**	53.22%	-3.427%
	H **	**61.89%!**	58.53%	-3.357%
	FT **	**62.83%!**	59.02%	-3.811%
	FH *	63.6%!	61.4%	-2.203%
Catalan	S =	26.19%!	25.83%	-0.3547%
	T **	**47.06%!**	45.24%	-1.827%
	H **	**65.88%!**	56.46%	-9.427%
	FT **	**66.02%!**	56.7%	-9.321%
	FH **	**69.8%!**	63.02%	-6.784%
Spanish	S **	**26.73%!**	23.79%	-2.943%
	T **	**47.16%!**	44.99%	-2.173%
	H **	**66.31%!**	57.44%	-8.871%
	FT **	**65.1%!**	55.72%	-9.378%
	FH **	**70.33%!**	62.83%	-7.501%
Czech	S **	**35.17%!**	32.07%	-3.1%
	T **	**59.36%!**	55.73%	-3.636%
	H **	**69.13%!**	62.57%	-6.562%
	FT **	**64.29%!**	59.51%	-4.783%
	FH **	**68.88%!**	63.8%	-5.081%
o-Czech	S **	**35.18%!**	32.47%	-2.708%
	T **	**59.01%!**	55.99%	-3.025%
	H **	**69.04%!**	62.64%	-6.404%
	FT **	**64.11%!**	60.02%	-4.091%
	FH **	**68.9%!**	63.68%	-5.222%
Japanese	S *	35.47%!	33.61%	-1.862%
	T **	**55.97%!**	51.95%	-4.025%
	H **	**71.32%!**	65.6%	-5.718%
	FT **	**59.64%!**	55.78%	-3.856%
	FH **	**70.13%!**	65.21%	-4.928%
total	S	*9, **8	*0, **0	-4.382%
	T	*10, **10	*0, **0	-4.574%
	H	*10, **10	*0, **0	-8.113%
	FT	*10, **10	*0, **0	-7.758%
	FH	*10, **9	*0, **0	-6.342%
	all	*49, **47	*0 ,**0	-6.234%

5.4.2 Dependency Relations

In the previous experiment the nodes of the linear sub-trees kept the dependency relation to their head word, but without knowing which one was the head word. Therefore, the drop in accuracy was not generated by the loss of the dependency relation label, however the dependency relation label lost part of its meaning because the nodes lost the head word.

So the next experiment replicates the previous one but without dependency relations. Table 5.9 shows the results, showing similar drops in accuracy, reinforcing the same conclusions as before: structural information is fundamental for achieving high accuracy.

5.5 Conclusion

This chapter presented the main experimental results, it starts presenting three base lines: the majority label, the Shape stings, and the binary settings. Majority label and Shape do not use any lexical or syntactic feature, for that reason it was expected similar performance across them. However, the results show that Shape outperforms the Majority label base line, which implies that the structural information provided by the syntactic parser is very important for the argument classification task. It was confirmed at the end of the chapter by training the system with the structural information and without it (string representation) and concluding that the system loses around 6% accuracy in average when structural information is removed. In regard to the comparison across the four settings: Ternary, Hamming, Frame Ternary and Frame Hamming, it was observed that as the complexity of the settings increases, the system is capable to achieve higher accuracy, with a few exceptions.

The Japanese data set reported the worst in-domain results, which is probably due to a poor annotation of the dependency relations. The outcomes of Spanish and Catalan are very similar to each other, which is probably due to the fact that most of their content is a translation of each other.

The analysis effect of the frame mismatch cost shows that having a frame mismatch cost is beneficial for the system as long as it does not ban samples from other frames, the value of the optimal cost will be analyzed in further detail at the Chapter 6. It is an interesting observation that the Chinese and German data produced the worse results in the CoNLL-2009 public evaluation but the best results on this system, it could imply that this system is more suitable for those languages or it could imply that argument identification was a harder task on those languages. One of the final analyses pointed out that the system is more accurate predicting smaller sub-trees (predicates with low amount of arguments).

Chapter 6

Further parameter refinement

6.1 Introduction

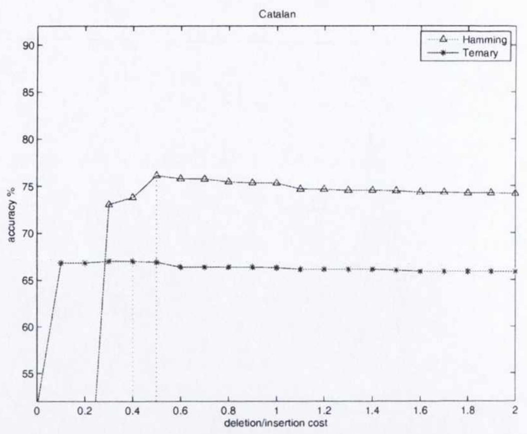
This chapter is an attempt to optimize the parameters that had been used in the previous one by default. In particular the parameters about: deletion costs, weight between POS and Dependency Relations, weight between lexical and syntactic features, optimal value for the k of the k-NN module and optimal value for the frame mismatch cost.

As the goal of this chapter is to optimize parameters, the system will be evaluated with the development data set, because the intention is to re-evaluate the system using those optimal parameters on the evaluation data set.[1] This intention makes the evaluation data set ineligible for the tuning. Because when a data set is used for tuning indirectly it becomes part of the training data, (as it was used to select the parameters). Training data should not be used to evaluate the system because it creates the illusion of a higher accuracy than it really performs.

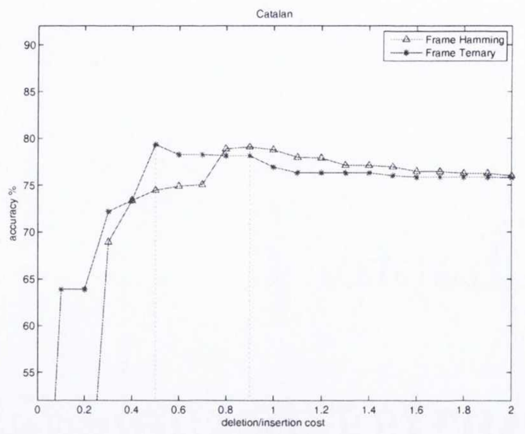
6.2 Varying Deletion cost

The following graphs will show the accuracy of Ternary, Hamming, Frame Ternary and Hamming swapping settings across different deletion/insertion costs per each language:

[1]Details on the development data set can be found at Section 2.4.2 on page 22.



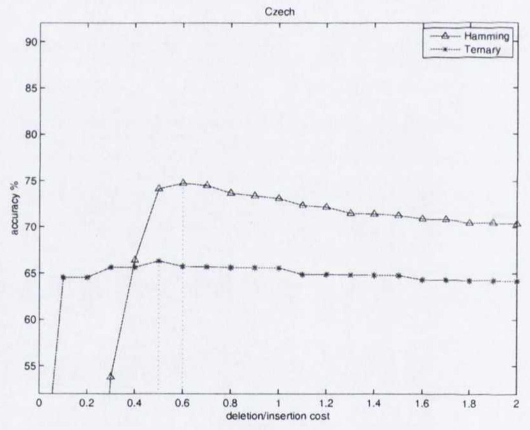
(a) simple



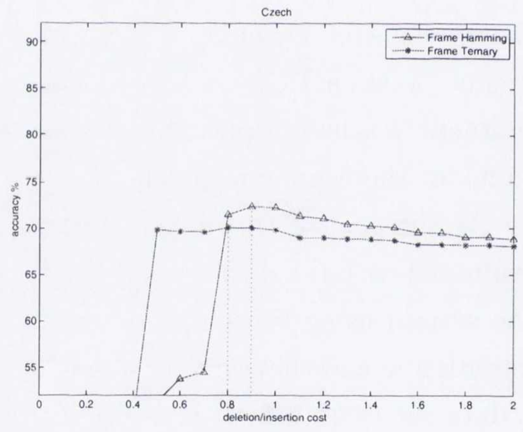
(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	1.726	18.27	33.37	68.93	73.37	74.45	74.87	75.04	78.88	79.07	78.77	77.96
FT	18.98	63.89	63.89	72.21	73.37	79.33	78.26	78.26	78.12	78.12	76.91	76.31
H	1.839	18.9	34.32	73.04	73.8	76.1	75.78	75.74	75.43	75.33	75.29	74.66
T	51.3	66.83	66.83	67.01	67.01	66.89	66.38	66.38	66.37	66.37	66.31	66.13

Figure 6.1: Tuning deletion cost in Catalan



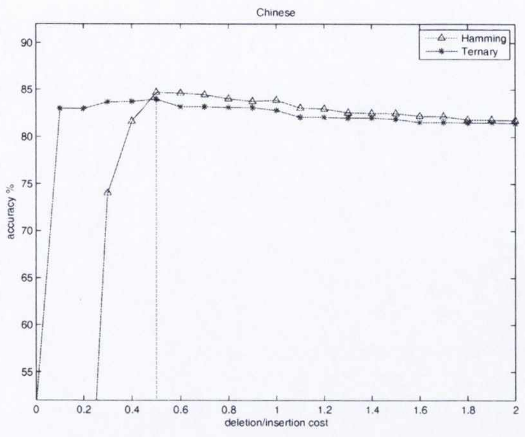
(a) simple



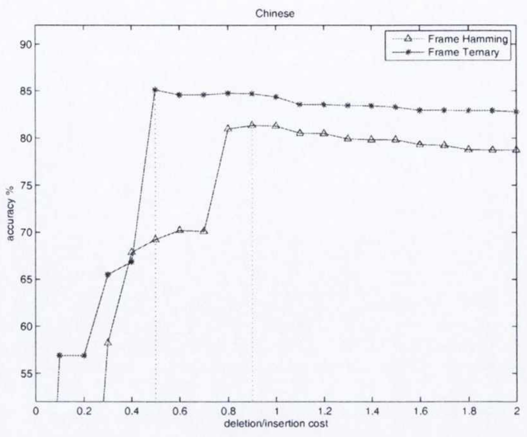
(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	0.1549	5.606	10.89	36.62	48.13	51.28	53.74	54.5	71.45	72.32	72.22	71.33
FT	3.456	34.42	34.42	47.53	49.06	69.82	69.65	69.6	70.04	70.04	69.81	68.97
H	0.1569	5.814	11.3	53.8	66.38	74.12	74.71	74.47	73.65	73.36	73.08	72.33
T	36.72	64.53	64.53	65.62	65.61	66.31	65.69	65.66	65.58	65.56	65.56	64.86

Figure 6.3: Tuning deletion cost in Czech



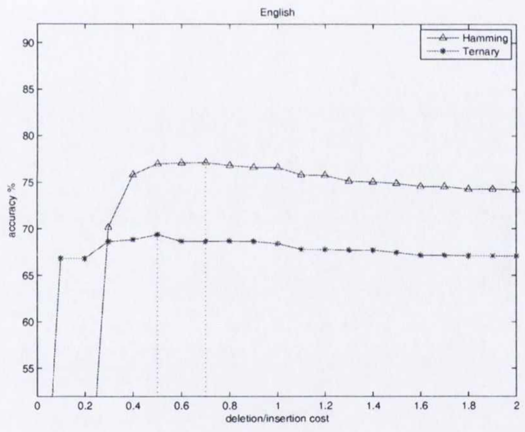
(a) simple



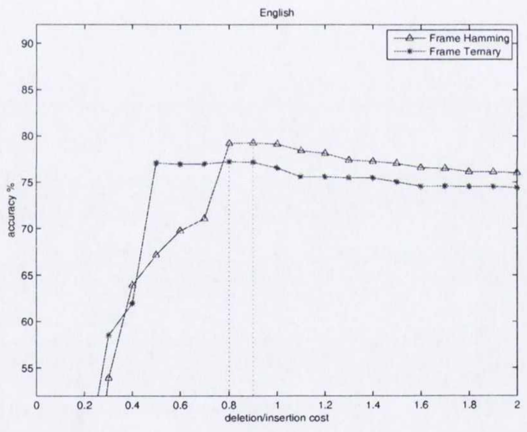
(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	0.9863	20.86	26.81	58.21	67.86	69.23	70.21	70.09	80.97	81.34	81.28	80.51
FT	9.469	56.89	56.89	65.5	66.85	85.11	84.58	84.58	84.75	84.7	84.35	83.56
H	1.029	21.37	27.51	74.02	81.64	84.68	84.61	84.46	84.03	83.75	83.88	83.05
T	51.28	82.96	82.96	83.68	83.73	83.96	83.17	83.17	83.11	83.09	82.8	82.08

Figure 6.2: Tuning deletion cost in Chinese



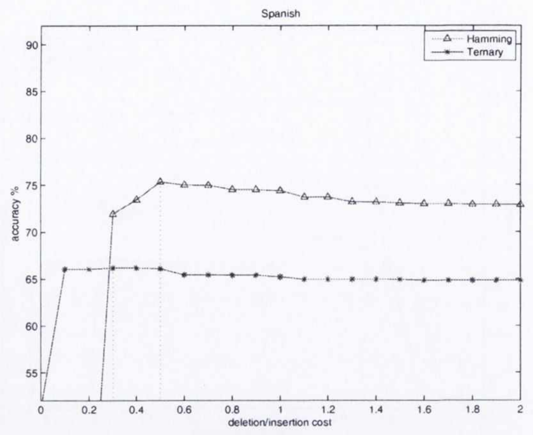
(a) simple



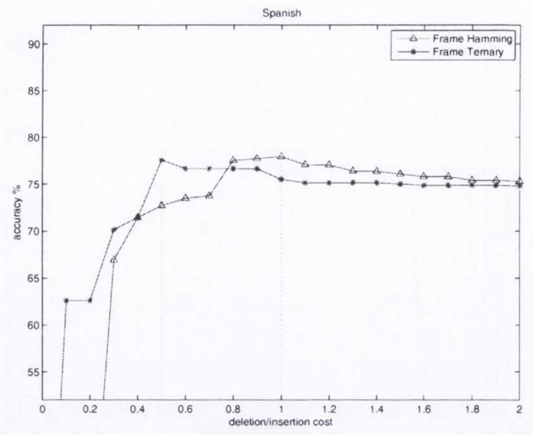
(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	2.192	26.38	33.89	53.95	63.91	67.18	69.77	71.06	79.18	79.24	79.11	78.39
FT	5.459	44.05	44.05	58.55	61.96	77.07	76.94	76.97	77.2	77.15	76.55	75.58
H	2.2	27.03	34.65	70.16	75.81	76.96	77.05	77.09	76.79	76.59	76.58	75.8
T	26.63	66.81	66.81	68.64	68.83	69.36	68.64	68.64	68.66	68.64	68.4	67.78

Figure 6.4: Tuning deletion cost in English



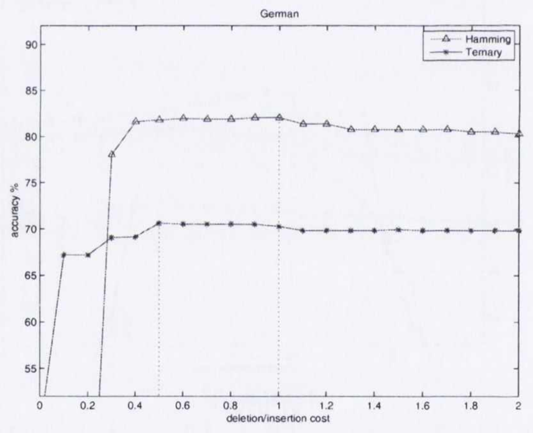
(a) simple



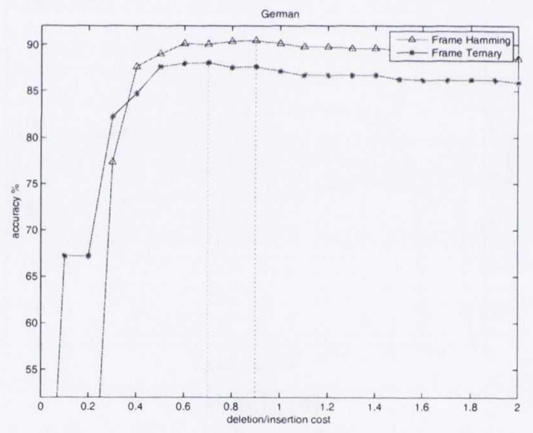
(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	0.9741	17.15	32.26	66.93	71.44	72.71	73.47	73.74	77.5	77.7	77.92	77.04
FT	16.99	62.59	62.59	70.16	71.42	77.56	76.61	76.61	76.61	76.61	75.48	75.11
H	1.129	17.94	33.1	71.92	73.43	75.32	74.99	74.96	74.48	74.48	74.37	73.67
T	51.06	66.05	66.05	66.19	66.18	66.09	65.43	65.43	65.39	65.39	65.2	64.94

Figure 6.5: Tuning deletion cost in Spanish



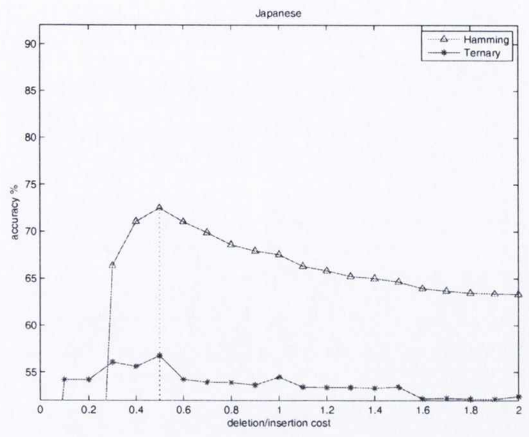
(a) simple



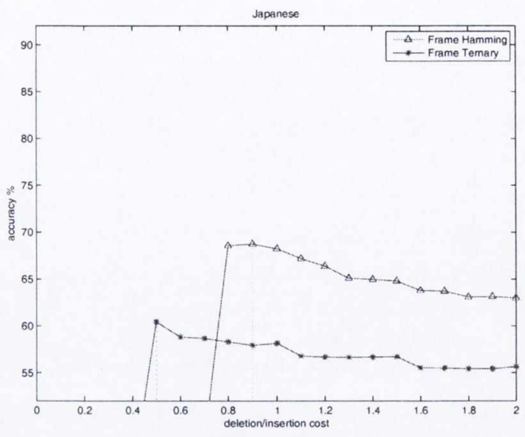
(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	3.162	17.78	26.75	77.35	87.61	88.97	90.09	90	90.34	90.43	90.09	89.74
FT	18.12	67.18	67.18	82.22	84.7	87.61	87.95	88.03	87.52	87.61	87.09	86.67
H	3.162	18.46	27.52	78.03	81.62	81.79	81.97	81.88	81.88	82.05	82.05	81.37
T	47.95	67.18	67.18	69.06	69.15	70.6	70.51	70.51	70.51	70.51	70.26	69.83

Figure 6.6: Tuning deletion cost in German



(a) simple



(b) frame

cost:	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
FH	1.034	13.63	14.96	38.23	46.4	47.51	48.06	47.73	68.56	68.75	68.23	67.2
FT	5.726	37.42	37.42	43.11	43.96	60.44	58.81	58.66	58.29	57.92	58.15	56.78
H	1.034	13.59	15	66.35	71.07	72.48	71.04	69.89	68.6	67.93	67.57	66.31
T	29.52	54.19	54.19	56.08	55.6	56.74	54.23	53.93	53.9	53.64	54.49	53.42

Figure 6.7: Tuning deletion cost in Japanese

Table 6.1: Dataset split by the maximum accuracy for different deletion costs

This table graphically summarizes were the maximum accuracy on the development data set for different deletion/insertion costs values was found. The column Max, specifies at which deletion value the high performance was found, the remaining other four columns correspond to the four atomic settings. Hamming and Frame Ternary seem to behave in a very similar manner.
For instance for Ternary settings at deletion cost 0.5 yield the maximum accuracy scores for German, Japanese English, Chinese, and Czech development data set.

Max	Ternary	Hamming	Frame Ternary	Frame Hamming
0.1				
0.2				
0.3	es			
0.4	ca			
0.5	ge,ja,en,chi,cz	ja,es,ch,ca	ja,es,ch,ca	
0.6		cz		
0.7		en	ge	
0.8			en,cz	cz
0.9				ge,ja,en,ch,ca
1		ge		es
macro avg	0.45	0.614	0.614	0.9

Deletion costs under zero lead the tree distance system to delete all nodes, thus deleting all nodes will have a negative distance which will be less than swapping all nodes at zero cost. Therefore, for any deletion cost under zero will lead to zero accuracy.

As deletion costs get near zero the accuracy drops dramatically, at deletion costs equal to zero the accuracy is above zero because the implementation to produce the alignment have preference for swaps than for deletion and insertion when the cost is the equivalent.

A first glance is that Frame versions tend to start falling at higher deletion values than the simple versions (as deletion cost becomes smaller)

Simple Ternary in general is very stable across different deletion values in comparison with the other three measures.

6.2.1 Ternary

This section will explain the details of the trends on Ternary setting as deletion cost changes from the previous graphs.

As the deletion cost become lower, some type of swaps become impossible because the cost of deleting and inserting become less costly than the cost of swapping them. Those singular points are analysed in the following subsections.

6.2.1.1 Both labels are different

Table 6.2: Deletion 0.4 vs 0.5: swapping unrelated nodes: Ternary

accuracy		del=.4	del=.5	difference
Chinese	T *	83.73%	83.96%!	0.2264%
German	T *	69.15%	70.6%!	1.453%
English	T *	68.83%	69.36%!	0.5337%
Catalan	T =	67.01%!	66.89%	-0.1214%
Spanish	T =	66.18%!	66.09%	-0.09482%
Czech	T **	65.61%	**66.31%!**	0.6929%
Japanese	T *	55.6%	56.74%!	1.145%
total	T	*0, **0	*5, **1	0.5478%

In this case for Ternary the swapping of two nodes with both different labels to each other have a cost of 1. Therefore, this kind of swaps are not allow for deletion values under 0.5. Hence, it should be observable the effect between deletion cost 0.4 and 0.5. Table 6.2 contrast the results for both deletion values.

Making Ternary not able to swap unrelated nodes (0.4) reduces accuracy; it is shown to be strictly significant different only for Czech, and less strictly significant different for five languages, but the average difference is as small as

0.55%. Spanish and Catalan score better without allowing that kind of swaps (deletion =0.4).

The conclusion is that the swaps between unrelated nodes have almost no impact on the system performance.

6.2.1.2 One label is different

Table 6.3: Deletion 0.2 vs 0.3: one label is different: Ternary

accuracy		del=.2	del=.3	difference
Chinese	T **	82.96%	**83.68%!**	0.7168%
German	T **	67.18%	**69.06%!**	1.88%
English	T **	66.81%	**68.64%!**	1.832%
Catalan	T *	66.83%	67.01%!	0.1735%
Spanish	T =	66.05%	66.19%!	0.1465%
Czech	T **	64.53%	**65.62%!**	1.09%
Japanese	T **	54.19%	**56.08%!**	1.884%
total	T	*0, **0	*6, **5	1.103%

If one label is different and the other is equal, the atomic swapping cost will be 0.5. Therefore, this kind of swap are not allowed for deletion values under 0.25. Hence, it should be observable the effect between deletion cost 0.2 and 0.3. Table 6.3 contrast the results for both deletion values.

Making Ternary not able to swap nodes with one different label (0.2) always has a negative impact, and it is shown to be strictly significantly different for five languages in spite of that the average differences is only around 1.1%. It means that this kind of swap also not very important for the performance of the system. This explains why Binary and Ternary measures perform in so similar way; See Figure 5.3 at page 80.

6.2.1.3 Ternary equality for 0.1 and 0.2

For Ternary the differences on accuracy between deletion/insertion cost =0.1 and 0.2 are exactly zero. This is visible as a horizontal line in all Ternary plots. This phenomena is generated because the alignments and the ranking of alignments are identical. The alignments are equivalent because the less costly swap in Ternary is 0.5 and in both deletion settings the cost of deleting and inserting two nodes is less than 0.5 (0.2 and 0.4). It implies that nodes will be only swept if *POS* and *DEPREL* labels are identical at zero cost. Hence, the cost of an alignment will be dependent only on the deletion/insertion cost. Hence, the ranking of neighbours alignments will be the same, but the distance will be scaled because the alignments are the same and the only cost of the alignments are the deletion and insertion costs which in both cases are positive.

For instance, if an alignment contains 7 deletions/insertions and one perfect match (it can not contain imperfect matches as described before). The cost of the alignment will be 0.7 (for deletion =0.1) and 1.4 (for deletion=0.2). For any amount of deletions/insertions the cost will be 0.1*x and 0.2*x, which keeps any alignment in the same order.

6.2.1.4 Both labels are equal

Table 6.4: Deletion 0 vs 0.1: ranking effect: Ternary

accuracy		del=0	del=0.1	difference
Chinese	T **	51.28%	**82.96%!**	31.68%
German	T **	47.95%	**67.18%!**	19.23%
English	T **	26.63%	**66.81%!**	40.18%
Catalan	T **	51.3%	**66.83%!**	15.53%
Spanish	T **	51.06%	**66.05%!**	14.99%
Czech	T **	36.72%	**64.53%!**	27.8%
Japanese	T **	29.52%	**54.19%!**	24.68%
total	T	*0, **0	*7, **7	24.87%

For values 0, 0.1 and 0.2 the tree edit distance implementation will generate the same alignment for two given trees. The results for 0 and 0.1 are different because for deletion cost equal zero any alignment for two given trees has a distance zero. Therefore, there is no ranking of alignments for deletion = 0, and that is what makes the accuracy to drop. Table 6.4 contrast the results for both deletion values. With ranking the system always performs strictly significantly better with a large margin of difference of 25%.

It is important to note that the system makes an approximation by taking the only first thousand samples into account and if the first equivalence class has more than a thousand samples it will be like removing random samples until the class only contains one thousand. In the case of deletion cost equal zero, as all samples are at the same distance, it should be considered that the selection was random of the thousand used samples was random.

6.2.2 Frame Ternary

This section will explain the details of the trends on Frame Ternary settings as deletion costs change. Like in Ternary, as the deletion cost become lower, some type of swaps become impossible because the cost of deleting and inserting become less costly than the cost of swapping them. Those singular points are analyse in text that follows.

6.2.2.1 In the predicate both labels are different: 0.9 and 1

Table 6.5: Estimated percentage of atomic swaps per language for Frame Ternary for Predicate Nodes which belong to different frame

POS	DepRel	Chinese	German	English	Catalan	Spanish	Czech	Japanese
=	=	27.52	24.32	5.14	27.52	27.07	7.59	23.42
<>	=	4.31	9.79	6.34	4.31	3.73	13.15	75.23
=	<>	58.42	17.11	13.98	58.42	60.38	21.20	0.28
<>	<>	9.75	48.78	74.54	9.75	8.83	58.07	1.07

Table 6.5 shows the probability for two random predicate nodes which belong to a different frame to have POS or DepRel labels equal or different.

Table 6.5 reads as follows: symbol <> in the column POS means that in this particular row the label POS is different from one node to another, symbol = indicates that POS label are identical for both nodes. The same can be said about the DepRel column. In the case of Chinese 9.75% of possible swaps differ in both labels. The table shows the percentage of swaps that will be banned as the deletion cost will decreases, because if one deletion and one insertion cost less than a swap, that type of swaps will never happen.âĂİ

Table 6.6: Deletion 0.9 vs 1: Frame Ternary

accuracy		del=.9	del=1.0	difference
Chinese	FT **	**84.7%!**	84.35%	-0.3449%
German	FT =	87.61%!	87.09%	-0.5128%
English	FT **	**77.15%!**	76.55%	-0.5986%
Catalan	FT **	**78.12%!**	76.91%	-1.206%
Spanish	FT **	**76.61%!**	75.48%	-1.129%
Czech	FT **	**70.04%!**	69.81%	-0.2262%
Japanese	FT =	57.92%	58.15%!	0.2216%
total	FT	*5, **5	*0, **0	-0.5422%

For Frame Ternary if there is a mismatch on the frame of the predicate node and both labels on the predicate node are different the swapping cost will be 2. Therefore, it should be observable a change between 0.9 and 1.

Table 6.6 contrasts the results for both deletion values. Allowing this kind of swap between unrelated predicate nodes have negative impact, but the differences are very small even though the difference was highly significantly significant in five languages. It is not a surprise that the differences are very small; as shown in Table 6.5 last row, this kind of swaps are quite rare for most of the languages even by making the swaps by random nodes, with English and Czech being the only exceptions where half of the swaps would have both different labels.

Table 6.5 do not intend to measure how difficult is task across the different languages but to measure who many possible swaps will be banned by changing the cost settings.

6.2.2.2 In the predicate one label is different: 0.7 and 0.8

Table 6.7: Deletion 0.7 vs 0.8: Frame Ternary

accuracy		del=0.7	del=0.8	difference
Chinese	FT *	84.58%	84.75%!	0.1725%
German	FT =	88.03%!	87.52%	-0.5128%
English	FT =	76.97%	77.2%!	0.2236%
Catalan	FT =	78.26%!	78.12%	-0.1388%
Spanish	FT =	76.61%	76.61%!	0%
Czech	FT **	69.6%	**70.04%!**	**0.4463%**
Japanese	FT =	58.66%!	58.29%	-0.3694%
total	FT	*0, **0	*2, **1	-0.02553%

For Frame Ternary if there is a mismatch on the predicate node and one of the labels is different the swapping cost will be 1.5. Therefore, it should be observable a change between 0.7 and 0.8. In 0.8 two predicates from different frames will be able to be mapped to each other even with one different label (POS or DepRel). Table 6.7 contrast the results for both deletion values.

Interestingly there is rarely a significant difference detected, it means that this kind of swaps between predicate nodes from different frames with one different label (POS or DepRel) are not very relevant to the performance of the system.

6.2.2.3 Both labels are different or in the predicate one is different

Table 6.8: Deletion 0.4 vs 0.5: swapping unrelated nodes: Frame Ternary

accuracy		del=.4	del=.5	difference
Chinese	FT **	66.85%	**85.11%!**	18.26%
German	FT *	84.7%	87.61%!	2.906%
English	FT **	61.96%	**77.07%!**	15.11%
Catalan	FT **	73.37%	**79.33%!**	5.967%
Spanish	FT **	71.42%	**77.56%!**	6.137%
Czech	FT **	49.06%	**69.82%!**	20.76%
Japanese	FT **	43.96%	**60.44%!**	16.48%
total	FT	*0, **0	*7, **6	12.23%

Even if POS and DepRel labels are equal the frame mismatch makes the atomic cost to be already 1. Therefore, swapping predicates of different frames will be not allowed for values under 0.5, also regular nodes with both labels different will be not allowed.

It has an important impact on the accuracy of the system. On average, the difference is 12.23%, always in favour of 0.5 for which 6 times was detected strictly significant difference.

This big drop was not detected for Ternary, which indicates that being able to use samples from different frames is important for the system performance. See Table 6.2 in page 114.

6.2.2.4 One label is different

Table 6.9: Deletion 0.2 vs 0.3: one label is different: Frame Ternary

accuracy		del=.2	del=.3	difference
Chinese	FT **	56.89%	**65.5%!**	8.612%
German	FT **	67.18%	**82.22%!**	15.04%
English	FT **	44.05%	**58.55%!**	14.5%
Catalan	FT **	63.89%	**72.21%!**	8.317%
Spanish	FT **	62.59%	**70.16%!**	7.568%
Czech	FT **	34.42%	**47.53%!**	13.11%
Japanese	FT **	37.42%	**43.11%!**	5.689%
total	FT	*0, **0	*7, **7	10.4%

In this case the swapping cost will be 0.5. Therefore, this kind of swaps is not allowed for deletion values under 0.25. Hence, it should be observable the effect between deletion cost 0.2 and 0.3. Table 6.9 contrast the results for both deletion values.

Making Frame Ternary not being able to swap nodes with one different label (0.2) always has an important negative impact and it is shown to be strictly significantly different for all languages, with an average drop of 10.4%.

6.2.2.5 Frame Ternary equality for 0.1 and 0.2

For Frame Ternary as in Ternary the differences on accuracy between deletion/insertion cost =0.1 and 0.2 are exactly zero. This phenomena is generated because the alignments and the ranking of alignments are identical. It was already explained for Ternary in Section 6.2.1.3 at page 115.

6.2.2.6 Both labels are equal

Table 6.10: Deletion 0 vs 0.1: ranking effect: Frame Ternary

accuracy		del=0	del=0.1	difference
Chinese	FT **	9.469%	**56.89%!**	47.42%
German	FT **	18.12%	**67.18%!**	49.06%
English	FT **	5.459%	**44.05%!**	38.59%
Catalan	FT **	18.98%	**63.89%!**	44.92%
Spanish	FT **	16.99%	**62.59%!**	45.6%
Czech	FT **	3.456%	**34.42%!**	30.97%
Japanese	FT **	5.726%	**37.42%!**	31.7%
total	FT	*0, **0	*7, **7	41.18%

The same described for Ternary applies to Frame Ternary: for values 0 and 0.1 the tree edit distance implementation will generate the same alignment for two given trees but for zero the ranking of alignments is lost.

Table 6.10 contrast the results for both deletion values. With ranking, the system always performs strictly significantly better with a large margin of difference of 41%.

6.2.3 Hamming

The cost of swapping on Hamming only can have five values depending on the amount of different labels between two nodes: 0,0.25,0.5, 0.75 and 1. it should be observable a singular point at each half of those values.

6.2.3.1 Perfect label mismatch

Table 6.11: Deletion 0.4 vs 0.5: Perfect label mismatch

accuracy		del=.4	del=.5	difference
Chinese	H **	81.64%	**84.68%!**	3.045%
German	H =	81.62%	81.79%!	0.1709%
English	H **	75.81%	**76.96%!**	1.147%
Catalan	H **	73.8%	**76.1%!**	2.298%
Spanish	H **	73.43%	**75.32%!**	1.888%
Czech	H **	66.38%	**74.12%!**	7.744%
Japanese	H *	71.07%	72.48%!	1.404%
total	H	*0, **0	*6, **5	2.528%

In case all labels were different the swapping cost will be 1. Hence, it should be observable a singular value at 0.5, because for values under 0.5 nodes with all different labels will be not be allowed to match each other. Table 6.11 contrast both values.

Except for Hamming on German language, for all other languages the fall is observable.

6.2.3.2 Three different labels

Table 6.12: Deletion 0.3 vs 0.4: Three different labels

accuracy		del=0.3	del=0.4	difference
Chinese	H **	74.02%	**81.64%!**	7.615%
German	H **	78.03%	**81.62%!**	3.59%
English	H **	70.16%	**75.81%!**	5.647%
Catalan	H **	73.04%	**73.8%!**	0.7546%
Spanish	H **	71.92%	**73.43%!**	1.508%
Czech	H **	53.8%	**66.38%!**	12.57%
Japanese	H **	66.35%	**71.07%!**	4.728%
total	H	*0, **0	*7, **7	5.202%

I expect that Form and Lemma are equal or different almost at the same ratio. Hence, at this point I expect that only POS or DepRel are equal, the singular value should be 0.375, under that value, this type of swaps will be not permitted. This should be observable comparing 0.3 and 0.4 values. Table 6.12 contrast both values.

For all languages a strict significant difference is detected with 5% difference in average. Hence, this type of swaps are important for the performance of the system.

6.2.3.3 Different form and lemma

Table 6.13: Deletion 0.2 vs 0.3:Different form and lemma

accuracy		del=.2	del=.3	difference
Chinese	H **	27.51%	**74.02%!**	46.52%
German	H **	27.52%	**78.03%!**	50.51%
English	H **	34.65%	**70.16%!**	35.51%
Catalan	H **	34.32%	**73.04%!**	38.73%
Spanish	H **	33.1%	**71.92%!**	38.82%
Czech	H **	11.3%	**53.8%!**	42.51%
Japanese	H **	15%	**66.35%!**	51.35%
total	H	*0, **0	*7, **7	43.42%

It is very likely that the lexical features will be different for most of the matches. Therefore, it is expected to find a singular value at 0.25, thus under this value the system will not map nodes with different labels for Lemma and Form. This should be observable comparing 0.2 and 0.3. Table 6.13 contrast both values. As expected the drop is very important: 43% in average and always with strictly significant difference.

6.2.3.4 One different label

Table 6.14: Deletion 0.1 vs 0.2:One different label

accuracy		del=.1	del=.2	difference
Chinese	H **	21.37%	**27.51%!**	6.133%
German	H **	18.46%	**27.52%!**	9.06%
English	H **	27.03%	**34.65%!**	7.623%
Catalan	H **	18.9%	**34.32%!**	15.42%
Spanish	H **	17.94%	**33.1%!**	15.16%
Czech	H **	5.814%	**11.3%!**	5.482%
Japanese	H **	13.59%	**15%!**	1.404%
total	H	*0, **0	*7, **7	8.612%

It should be another singular point at 1.25, because under this value, only perfect matched nodes will be allowed. It should be observable comparing 0.1 and 0.2. Table 6.14 contrast both values, as expected there is a significant difference for all languages with 8.6% difference in average.

6.2.3.5 Ranking

Table 6.15: Deletion 0 vs 0.1: Ranking

accuracy		del=0	del=.1	difference
Chinese	H **	1.029%	**21.37%!**	20.34%
German	H **	3.162%	**18.46%!**	15.3%
English	H **	2.2%	**27.03%!**	24.83%
Catalan	H **	1.839%	**18.9%!**	17.06%
Spanish	H **	1.129%	**17.94%!**	16.81%
Czech	H **	0.1569%	**5.814%!**	5.657%
Japanese	H **	1.034%	**13.59%!**	12.56%
total	H	*0, **0	*7, **7	16.08%

Like in Ternary the alignments at deletion cost 0 and 0.1 are the same, but at cost zero the ranking of alignments is lost. Table 6.15 shows contrast both values showing a really important lost of accuracy which leads to reach almost zero accuracy.

6.2.4 Frame Hamming

Frame Hamming will have the same singular points as Hamming plus a few more on the cases where predicates belong to different frames.

6.2.4.1 Frame mismatch + Perfect mismatch

Table 6.16: Deletion 0.9 vs 1: Frame mismatch + Perfect mismatch

accuracy		del=.9	del=1.0	difference
Chinese	FH =	81.34%!	81.28%	-0.05928%
German	FH =	90.43%!	90.09%	-0.3419%
English	FH =	79.24%!	79.11%	-0.1226%
Catalan	FH *	79.07%!	78.77%	-0.3036%
Spanish	FH =	77.7%	77.92%!	0.2155%
Czech	FH =	72.32%!	72.22%	-0.09782%
Japanese	FH =	68.75%!	68.23%	-0.5172%
total	FH	*1, **0	*0, **0	-0.1753%

It is unlikely for the system to swap two predicate nodes which do not share any label[2] and different frame but if it happens the effect of this swaps would be observable comparing deletion costs for 1 and 0.9, because for any value under 1 this kind of swaps are not permitted. Table 6.16 contrast both values.

[2]Remainder: each node contains for labels: Dependency relation, POS, Form and Lemma.

As expected no, significant difference is observed, not even for Spanish which shows its peak of performance on Frame Hamming at deletion cost 1.

6.2.4.2 Frame mismatch + Three label mismatch

Table 6.17: Deletion 0.8 vs 0.9:Frame mismatch + Three label mismatch

accuracy		del=.8	del=.9	difference
Chinese	FH *	80.97%	81.34%!	0.3665%
German	FH =	90.34%	90.43%!	0.08547%
English	FH =	79.18%	79.24%!	0.0577%
Catalan	FH =	78.88%	79.07%!	0.1908%
Spanish	FH =	77.5%	77.7%!	0.1983%
Czech	FH **	71.45%	**72.32%!**	0.8681%
Japanese	FH =	68.56%	68.75%!	0.1847%
total	FH	*0, **0	*2, **1	0.2788%

It is more likely for the system to swap two predicate nodes which differ on three labels. This kind of swaps are not permitted under a deletion cost equal to 0.875. Hence, it should be observable comparing 0.8 and 0.9. Table 6.17 contrast both values.

No important drop in accuracy was found in this step which is a surprise, because for most language the peak of performance was found at 0.9.

6.2.4.3 Frame mismatch + lexical labels

Table 6.18: Deletion 0.7 vs 0.8: Frame mismatch+lexical labels

accuracy		del=.7	del=.8	difference
Chinese	FH **	70.09%	**80.97%!**	10.88%
German	FH =	90%	90.34%!	0.3419%
English	FH **	71.06%	**79.18%!**	8.121%
Catalan	FH **	75.04%	**78.88%!**	3.842%
Spanish	FH **	73.74%	**77.5%!**	3.767%
Czech	FH **	54.5%	**71.45%!**	16.96%
Japanese	FH **	47.73%	**68.56%!**	20.83%
total	FH	*0, **0	*6, **6	9.248%

If two predicates belong to two different frames then they are very likely to differ also in Form and Lemma labels. Hence, the swapping cost will be 1.5, which predicts a singular value at 0.75. Under this value almost all samples from different frames are not allowed. Table 6.18 contrast 0.7 and 0.8 values.

As expected this kind of swaps have an important impact, showing an average of 9.24% improvement for cost 0.8 in Frame Hamming.

Table 6.19: Estimated percentage of atomic swaps per language for Hamming

Only the cases where lemma or form labels are identical are considered

Form	Lemma	Chinese	German	English	Catalan	Spanish	Czech	Japanese
=	<>	0.00	0.12	0.23	0.34	0.36	0.61	1.13
<>	=	0.00	**73.71**	13.45	48.03	38.56	15.63	5.49
=	=	100.00	26.17	86.32	51.62	61.08	83.76	93.39

The assumption of Form and Lemma labels will be likely to be different at the same time is applicable to all languages except German. Table 6.19 compares random nodes in which at least one of Form or Lemma labels are equal. In all languages except German, for two given nodes in which the Lemma label is identical the probability of also having an identical Form label is over 50%.

The few cases in which Form remains equal but Lemma remains different corresponds to ambiguous words such as left. For instance in the following two sentences of the evaluation data set:

(35) Many money managers and some traders had already **left** their offices early Friday afternoon on a warm autumn day – because the stock market was so quiet.

(36) And even if a nurse would wear flowers in her hair while on duty, if she were engaged she would know to wear them behind her **left**, not right, ear.

In Sentence 35 the word "left" has as lemma "leave" but in Sentence 36 the word "left" has as lemma "left" meaning location.

In other cases it seems that the divergence in form lemma correspond to differences between verbs and nouns forms:

(37) Then in a lightning plunge, the Dow Jones industrials in barely an hour surrendered about a third of their gains this year, chalking up a 190.58 - point, or 6.9 %, loss on the day in gargantuan **trading** volume.

(38) At 02:43 p.m. EDT , came the sickening news: The Big Board was halting **trading** in UAL , " pending news . "

In Sentence 37 the lemma for the word "trading" is also "trading" because it is a noun, but in Sentence 38 the lemma for the word "trading" is "trade" because it is a verb.

Regarding samples with equal Lemma but different Form, most of the cases are due to the inflexion of the language but some of them are also due to differences between upper and lower case characters depending if the word is the first one of the sentence or not.

Table 6.18 also shows Frame Hamming settings in German data set as the exception because no significant difference was detected.

6.2.4.4 Perfect label mismatch

Table 6.20: Deletion 0.4 vs 0.5: Perfect label mismatch: Frame Hamming

accuracy		del=.4	del=.5	difference
Chinese	FH **	67.86%	**69.23%!**	1.369%
German	FH *	87.61%	88.97%!	1.368%
English	FH **	63.91%	**67.18%!**	3.267%
Catalan	FH **	73.37%	**74.45%!**	1.084%
Spanish	FH **	71.44%	**72.71%!**	1.267%
Czech	FH **	48.13%	**51.28%!**	3.15%
Japanese	FH *	46.4%	47.51%!	1.108%
total	FH	*0, **0	*7, **5	1.802%

As in Hamming, in case all labels were different, the swapping cost will be 1. Hence, a singular value should be observable at 0.5. This should happen because for values under 0.5 nodes with all different labels will be not allow to match each other. Table 6.20 contrast both values. The differences are small(1.8%) but always was significant.

6.2.4.5 Three different labels

Table 6.21: Deletion 0.3 vs 0.4: Three different labels: Frame Hamming

accuracy		del=0.3	del=0.4	difference
Chinese	FH **	58.21%	**67.86%!**	9.647%
German	FH **	77.35%	**87.61%!**	10.26%
English	FH **	53.95%	**63.91%!**	9.96%
Catalan	FH **	68.93%	**73.37%!**	4.432%
Spanish	FH **	66.93%	**71.44%!**	4.517%
Czech	FH **	36.62%	**48.13%!**	11.51%
Japanese	FH **	38.23%	**46.4%!**	8.164%
total	FH	*0, **0	*7, **7	8.355%

As in Hamming, I expect that Form and Lemma are equal or different almost at the same time. Hence, at this point I expect that only POS or DepRel are equal, the singular value should be 0.375, under that value, this type of swaps will be not permitted. This should be observable comparing 0.3 and 0.4 values. Table 6.21 contrast both values.

For all languages a strictly significant difference is detected with a difference of 8% which is larger than the one detected for Hamming. Hence, this type of swap is important for the performance of the system.

6.2.4.6 Different form and lemma

Table 6.22: Deletion 0.2 vs 0.3:Different form and lemma: Frame Hamming

accuracy		del=.2	del=.3	difference
Chinese	FH **	26.81%	**58.21%!**	31.4%
German	FH **	26.75%	**77.35%!**	50.6%
English	FH **	33.89%	**53.95%!**	20.06%
Catalan	FH **	33.37%	**68.93%!**	35.56%
Spanish	FH **	32.26%	**66.93%!**	34.66%
Czech	FH **	10.89%	**36.62%!**	25.73%
Japanese	FH **	14.96%	**38.23%!**	23.27%
total	FH	*0, **0	*7, **7	31.61%

As mention before it is very likely that the lexical features will be different for most of the matches. As for Hamming it is expected to find a singular value at 0.25, thus under this value the system will not map nodes with different labels for Lemma and Form. This should be observable comparing 0.2 and 0.3. Table 6.22 contrast both values. As expected the drop is very important: 31% always with strictly significant difference but not as large as for Hamming (43%).

6.2.4.7 One different label

Table 6.23: Deletion 0.1 vs 0.2:One different label: Frame Hamming

accuracy		del=.1	del=.2	difference
Chinese	FH **	20.86%	**26.81%!**	5.95%
German	FH **	17.78%	**26.75%!**	8.974%
English	FH **	26.38%	**33.89%!**	7.508%
Catalan	FH **	18.27%	**33.37%!**	15.1%
Spanish	FH **	17.15%	**32.26%!**	15.12%
Czech	FH **	5.606%	**10.89%!**	5.282%
Japanese	FH **	13.63%	**14.96%!**	1.33%
total	FH	*0, **0	*7, **7	8.466%

There should be another singular point at 0.125, because under this value, only perfect matched nodes will be allowed. It should be observable comparing 0.1 and 0.2, which are contrasted in Table 6.23. In fact it was found for all languages a strict significant difference with an average loss of 8.5%.

6.2.4.8 Ranking

Table 6.24: Deletion 0 vs 0.1: Ranking: Frame Hamming

accuracy		del=0	del=.1	difference
Chinese	FH **	0.9863%	**20.86%!**	19.87%
German	FH **	3.162%	**17.78%!**	14.62%
English	FH **	2.192%	**26.38%!**	24.19%
Catalan	FH **	1.726%	**18.27%!**	16.55%
Spanish	FH **	0.9741%	**17.15%!**	16.17%
Czech	FH **	0.1549%	**5.606%!**	5.451%
Japanese	FH **	1.034%	**13.63%!**	12.6%
total	FH	*0, **0	*7, **7	15.63%

Like in Ternary, Frame Ternary and Hamming the alignments at deletion cost 0 and 0.1 are the same, but at cost zero the ranking of alignments is lost. Table 6.24 shows contrast both values showing a really important lost of accuracy to almost zero.

6.3 Weighting POS vs DepRel: Ternary and Frame Ternary

Ternary and Frame Ternary settings refer to POS and DepRel labels treating them as equally important. In this section it is consider weighting the cost

contribution of each of them. In particular it is introduced a parameter α which is used in cost definition as follows:
if POS is different add a cost of α,
and if DepRel is different add a cost of $1 - \alpha$.
Deletion and insertion cost remains equal to one.

For $\alpha = 0$ the system only uses Dependency Relation labels and ignores POS labels. For $\alpha = 1$ the system only uses POS labels and ignores Dependency relation labels.

The following graphs (from 6.8 to 6.13) show how the accuracy of Ternary and Frame ternary settings change in relation to α. There are seven graphs, one for each language data set, the x axis represent the different values of α and the y axis represents the accuracy of the system measured on the development data set. It seems intuitive that the optimal α value should be found between zero and one, because otherwise the system would be promoting swaps in which POS is different or the dependency relation are different. Intuition was tested by plotting the values between -.05 and 1.05. As expected, outside the range between zero and one, accuracy drops. The accuracy scale is the same in all graphs in order to make them easy to be compared to each other. Also, optimal values are highlighted by a vertical dotted line starting at the value of optimal accuracy to the bottom of the graph.

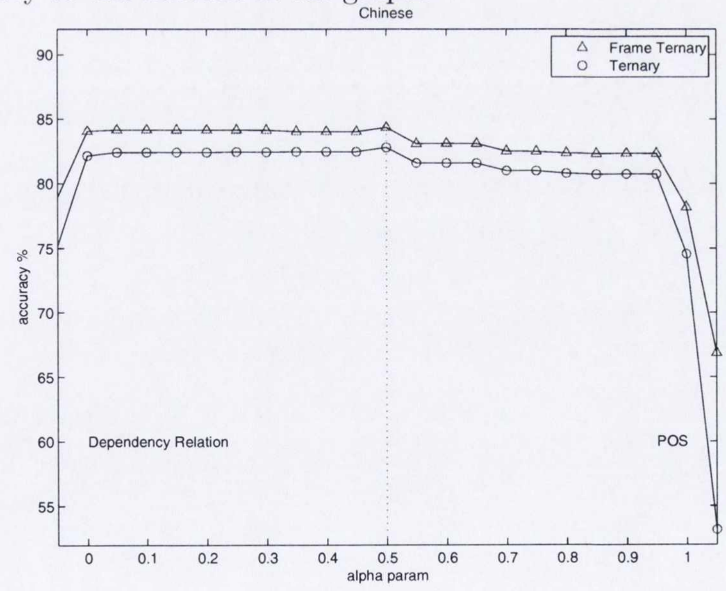


Figure 6.8: Tuning parameter alpha on Ternary Chinese development data

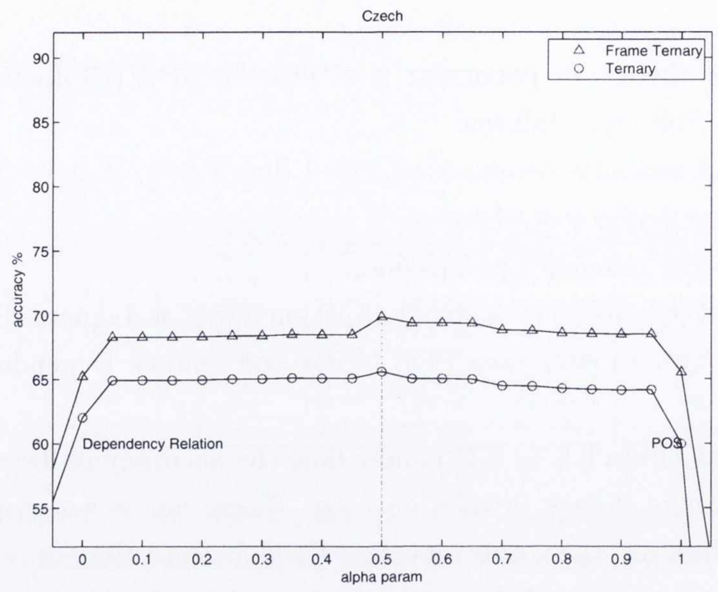


Figure 6.9: Tuning parameter alpha on Ternary Czech development data

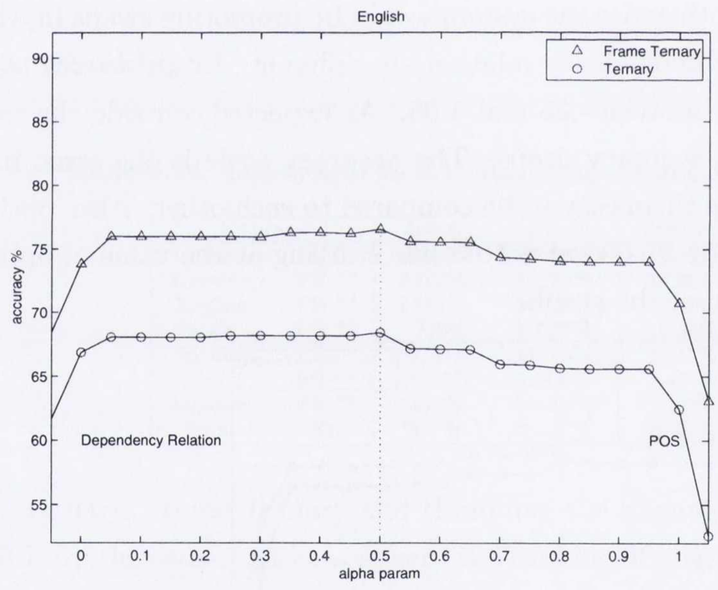


Figure 6.10: Tuning parameter alpha on Ternary English development data

Chinese, Czech and English data sets have in common that the maximum accuracy is reach with $\alpha = 1/2$, which is a very intuitive value and corresponds to Ternary system.

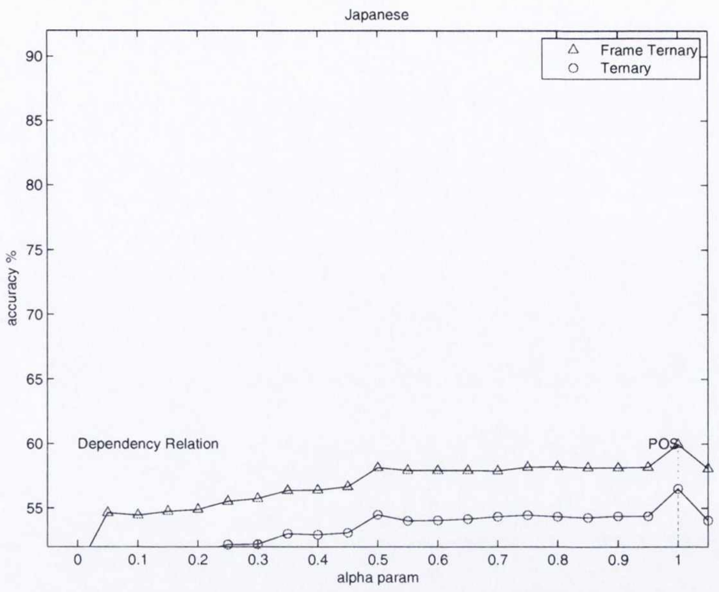


Figure 6.11: Tuning parameter alpha on Ternary Japanese development data

Japanese also has a local maximum of accuracy in $\alpha = 0.5$ for Ternary and Frame Ternary but the maximum accuracy is for $\alpha = 1$. The reason behind that is that there is almost no annotation on Dependency Relations.

It is interesting to note that in the public evaluation as shown in Table 5.6 on page 100, the systems perform fairly well in Japanese, in spite of the lack of dependency relation annotation.

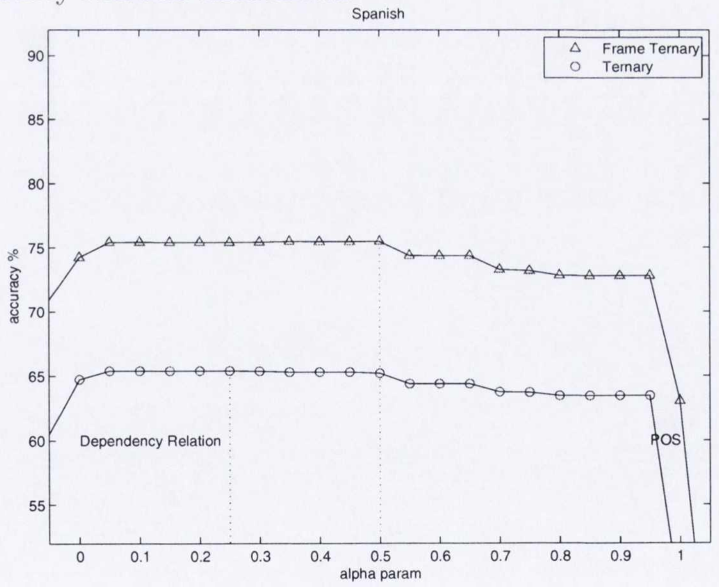


Figure 6.12: Tuning parameter alpha on Ternary Spanish development data

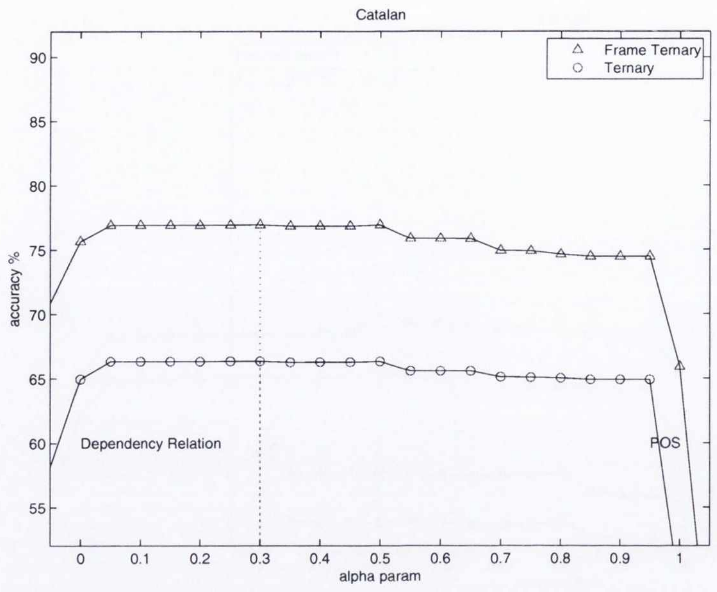


Figure 6.13: Tuning parameter alpha on Ternary Catalan development data

Spanish and Catalan show a flat crests of maximums with an inflexion point in $\alpha = 0.5$, over 0.5 the accuracy decreases. It happen for Ternary and Frame Ternary, but Catalan Frame Hamming seems to have the maximum value in $\alpha = 0.5$ but with a very small difference.

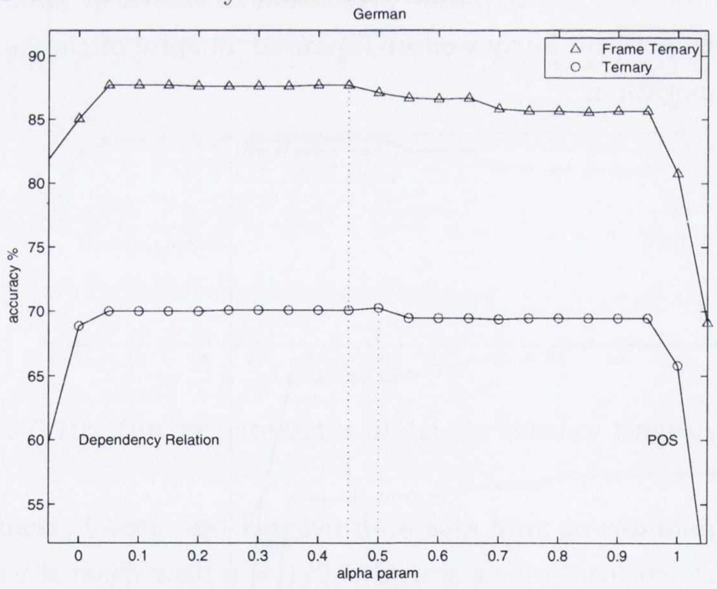


Figure 6.14: Tuning parameter alpha on Ternary German development data

The German graph looks very similar to Spanish and Catalan but it is unique in the sense that for Frame Ternary the maximum value is $\alpha = 0.45$, but Ternary shows the maximum value at $\alpha = 0.5$.

Conclusions

The parameter $\alpha = 1/2$ is not the optimal value for all languages but is optimal for most of the languages or very close the optimal value. Japanese is the most exceptional case because its data set lack annotation for the dependency relations what makes the optimal $\alpha = 1$, however still have a local maximum at 0.5. The lack of annotation dependency relations annotation seem to not affect the results in CoNLL-2009, in which Japanese is the second best predicted data in Table 100.

Table 6.25: Ternary comparing α extremes

accuracy		$\alpha = 0$	$\alpha = 1$	difference
Chinese	w-T **	**82.16%!**	74.57%	-7.583%
	w-FT **	**84.05%!**	78.17%	-5.885%
German	w-T *	68.89%!	65.81%	-3.077%
	w-FT **	**85.04%!**	80.77%	-4.274%
English	w-T **	**66.84%!**	62.44%	-4.399%
	w-FT **	**73.82%!**	70.76%	-3.065%
Catalan	w-T **	**64.95%!**	48.34%	-16.61%
	w-FT **	**75.65%!**	65.9%	-9.757%
Spanish	w-T **	**64.74%!**	46.88%	-17.87%
	w-FT **	**74.24%!**	63.05%	-11.2%
Czech	w-T **	**62.05%!**	59.98%	-2.064%
	w-FT =	65.22%	65.5%!	0.2751%
Japanese	w-T **	43.7%	**56.52%!**	12.82%
	w-FT **	50.24%	**59.96%!**	9.716%
total	w-T	*6, **5	*1, **1	-5.54%
	w-FT	*5, **5	*1, **1	-3.455%

Table 6.26: Ternary comparing α extremes limits

accuracy		$\alpha = 0.05$	$\alpha = 0.95$	difference
Chinese	w-T **	**82.41%!**	80.73%	-1.681%
	w-FT **	**84.16%!**	82.33%	-1.822%
German	w-T =	70%!	69.49%	-0.5128%
	w-FT *	87.69%!	85.64%	-2.051%
English	w-T **	**68.04%!**	65.61%	-2.43%
	w-FT **	**75.97%!**	73.99%	-1.976%
Catalan	w-T **	**66.32%!**	64.9%	-1.422%
	w-FT **	**76.92%!**	74.46%	-2.463%
Spanish	w-T **	**65.39%!**	63.47%	-1.922%
	w-FT **	**75.41%!**	72.8%	-2.612%
Czech	w-T **	**64.91%!**	64.16%	-0.756%
	w-FT =	68.28%	68.47%!	0.1916%
Japanese	w-T **	51.31%	**54.38%!**	3.066%
	w-FT **	54.64%	**58.18%!**	3.546%
total	w-T	*5, **5	*1, **1	-0.8085%
	w-FT	*5, **4	*1, **1	-1.027%

Table 6.27: Ternary comparing α patterns

αT	max accuracy	visual range maxims	
Chinese	0.5	0.5	0.5
German	0.5	0.05	0.5
English	0.5	0.5	0.5
Catalan	0.3	0.05	0.5
Spanish	0.25	0.05	0.5
Czech	0.5	0.5	0.5
Japanese	1	1	1
	Japanese has a local maximum at 0.5		

Table 6.28: Frame Ternary comparing α patterns

αFT	max accuracy	visual range maxims	
Chinese	0.5	0.5	0.5
German	0.45	0.05	0.45
English	0.5	0.5	0.5
Catalan	0.3	0.05	0.5
Spanish	0.5	0.05	0.5
Czech	0.5	0.5	0.5
Japanese	1	1	1
	Japanese has a local maximum at 0.5		

Tables 6.25 and 6.26 compares the accuracy of the system using dependency relations and using POS. For all languages (except Japanese) when the system use only dependency relations ($\alpha = 0$) the performance is better than when it uses only POS ($\alpha = 1$) (Table 6.25). Similar observations can be made by

observing the values near the extremes $\alpha = 0.05$ to $\alpha = 0.95$ (Table 6.26). The only exception is Czech Frame Ternary.

It suggests that Dependency Relations give more useful information than POS.

It is interesting that for all languages at least visually $\alpha = 0.5$ is within the set of optimal values. See Tables 6.27 and 6.28, with the single exception of German Frame Ternary.

The Japanese data set shows the maximum values at $\alpha = 1$ because the lack of Dependency Relation annotation, but surprisingly it was found a local maxims at $\alpha = 0.5$.

For all languages for $\alpha > 1$, the system penalizes to match nodes with equal dependency relations, as it was expected, the accuracy drops even more than when $\alpha = 1$.

Again, for all languages for $\alpha < 0$, the system penalized to match nodes with equal POS, as it was expected the accuracy always drops even more than with $\alpha = 0$.

6.4 Weighting lexical vs syntactic information: Hamming and Frame Hamming

Both Hamming and Frame Hamming refer to lexical (Lemma and Form labels) and syntactic (POS and DepRel) labels, treating both sets of labels as equally important. In this section it is consider different contributions for both sets of labels. It is introduced a parameter α to the Hamming and Frame Hamming measure in order to weigh the lexical features against the syntactic ones, so the swapping cost of two nodes is defined as:

- if POS is different add a cost of $\alpha/2$,
- if DepRel is different add an extra cost of $\alpha/2$.
- if Lemma is different add an extra cost of $(1-\alpha)/2$.
- if Form is different add an extra cost of $(1-\alpha)/2$.

For $\alpha = 0$ the system only uses Lemma and Form and ignores POS and DepRel labels. For $\alpha = 1$ the system become Ternary/Frame Ternary because it only uses POS and DepRel labels with the same weight as Ternary/Frame Ternary settings use it.

The following graphs (from 6.15 to 6.21) show how the accuracy of Hamming and Frame Hamming settings change in relation to α. There are seven graphs, one for each language data set, the x axis represent the different values of α and the y axis represents the accuracy of the system measured on the development data set. It seems intuitive that the optimal α value should be found between zero and one, because otherwise the system would be promoting swaps in which POS is different or the dependency relation are different. Intuition was tested by plotting the values between -.05 and 1.05. As expected, like in the experiments on the Ternary settings, outside the range between zero and one, accuracy drops. The accuracy scale is the same in all graphs in order to make them easy to be compared to each other. Also, optimal values are highlighted by a vertical dotted line starting at the value of optimal accuracy to the bottom of the graph.

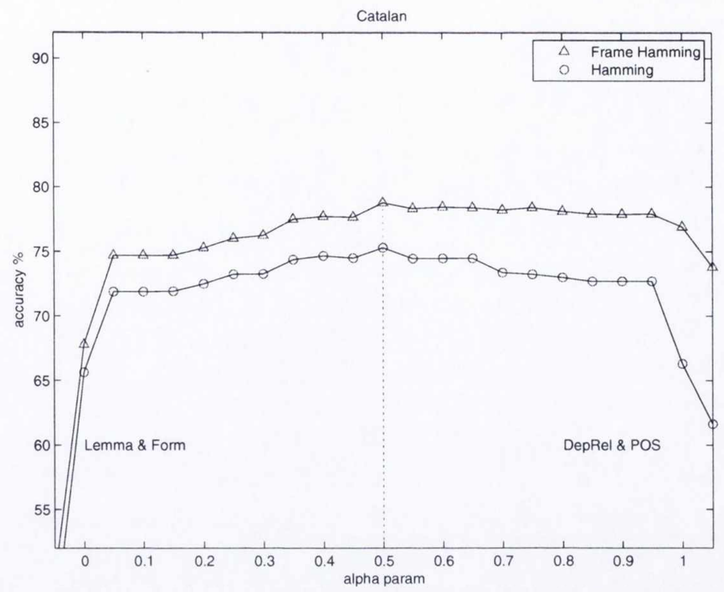


Figure 6.15: Tuning parameter alpha on Hamming Catalan development data

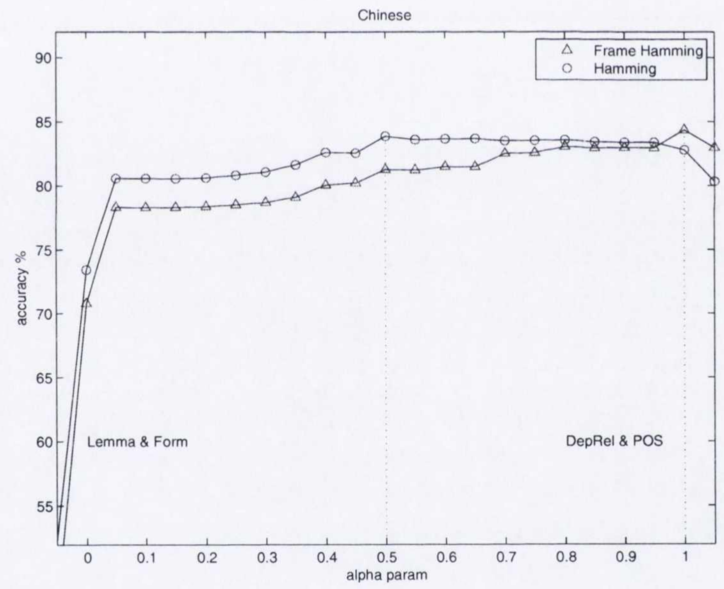


Figure 6.16: Tuning parameter alpha on Hamming Chinese development data

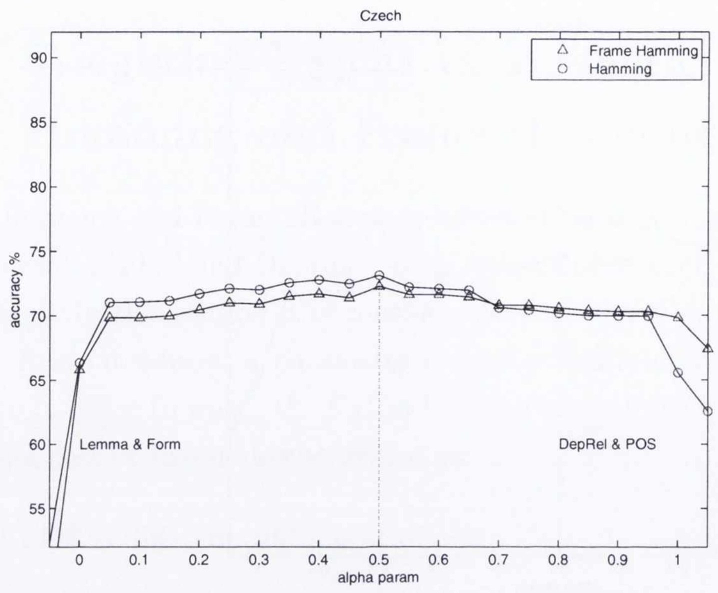


Figure 6.17: Tuning parameter alpha on Hamming Czech development data

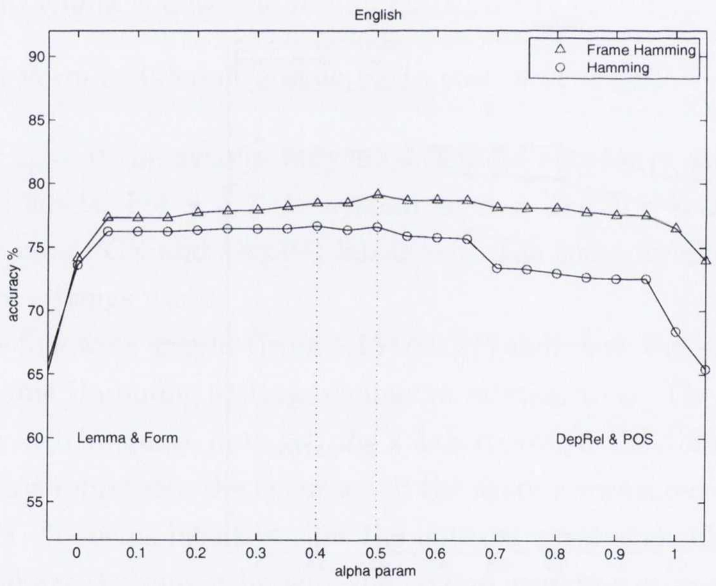


Figure 6.18: Tuning parameter alpha on Hamming English development data

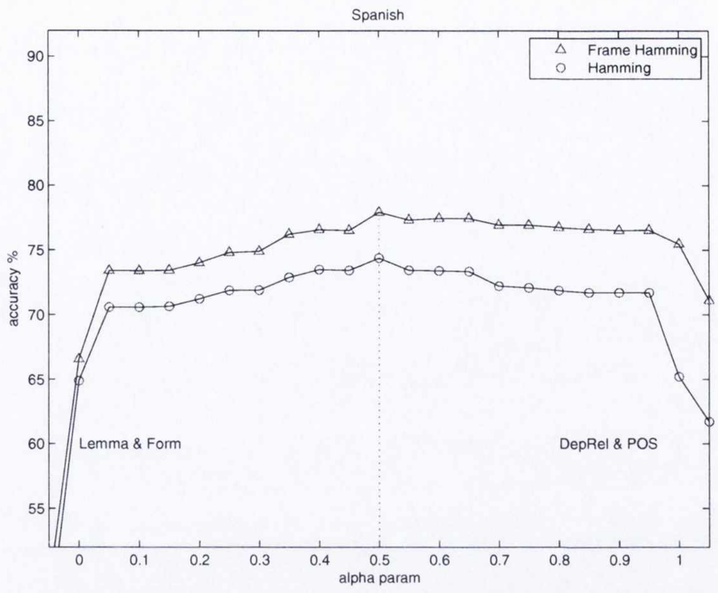


Figure 6.19: Tuning parameter alpha on Hamming Spanish development data

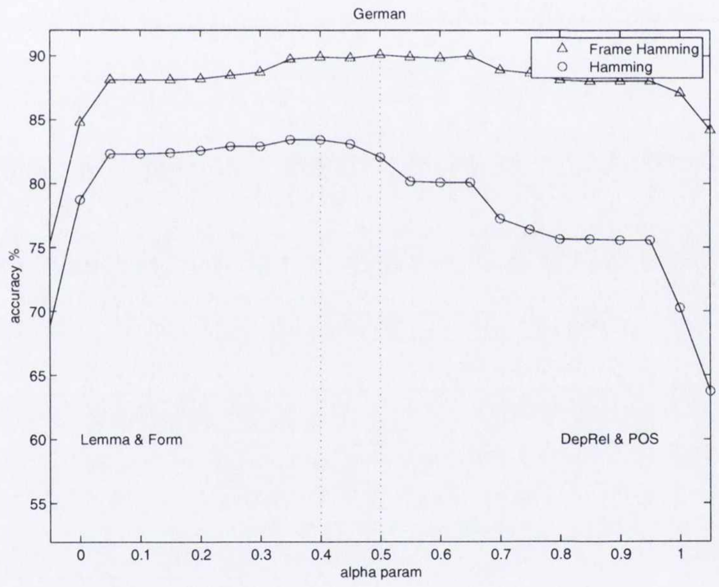


Figure 6.20: Tuning parameter alpha on Hamming German development data

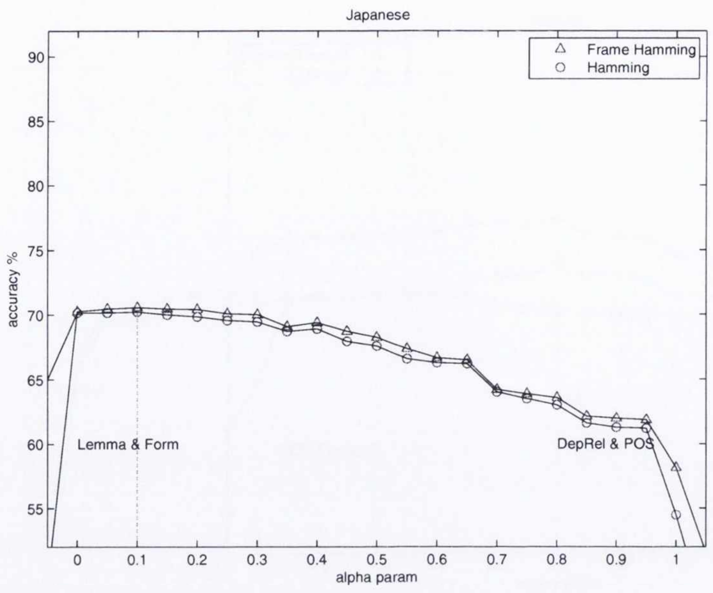


Figure 6.21: Tuning parameter alpha on Hamming Japanese development data

Conclusions

Table 6.29: Alpha values for maximum accuracy for each language on weighted Hamming and Frame Hamming

Each tag language is placed in the level in which its maximum accuracy was found, for instance the optimal value for alpha en the case of Hamming in English is 0.4.

max	H	FH
0		
0.1	jp	jp
0.2		
0.3		
0.4	en,ge	
0.5	ca,es,ch,cz	ca,es,cz,en,ge
0.6		
0.7		
0.8		
0.9		
1		ch

Table 6.29 shows for which value of α the maximum accuracy was found in each language for Hamming and Frame Hamming. For almost all cases 0.5 was the optimal value, which corresponds to the original definition of Hamming settings without α, in two occasions, the maximum value for Hamming is 0.4 which still is very close to 0.5. The two great exceptions to this trend are Japanese and Frame Hamming Chinese.

In the case of Chinese, as shown before, Frame Hamming performed worse than any other measure, so it found a way to become identical to Frame Ternary by choosing $\alpha = 1$. In the case of Japanese, the annotation of Dependency Relations is very poor, it was expected that the system would tend to give more value to Form and Lemma labels, what is unexpected is that the weight for lexical features is nine times the weight of the syntactic ones.

Table 6.30: Hamming comparing α extremes

accuracy		$\alpha = 0$	$\alpha = 1$	difference
Chinese	w-H **	73.44%	**82.8%!**	9.361%
	w-FH **	70.8%	**84.35%!**	13.55%
German	w-H **	**78.72%!**	70.26%	-8.462%
	w-FH *	84.79%	87.09%!	2.308%
English	w-H **	**73.58%!**	68.4%	-5.178%
	w-FH **	74.15%	**76.55%!**	2.394%
Catalan	w-H =	65.62%	66.31%!	0.6852%
	w-FH **	67.78%	**76.91%!**	9.133%
Spanish	w-H =	64.88%	65.2%!	0.3189%
	w-FH **	66.57%	**75.48%!**	8.904%
Czech	w-H *	66.36%!	65.56%	-0.7988%
	w-FH **	65.82%	**69.81%!**	3.986%
Japanese	w-H **	**70.15%!**	54.49%	-15.66%
	w-FH **	**70.26%!**	58.15%	-12.12%
total	w-H	*4, **3	*1, **1	-2.819%
	w-FH	*1, **1	*6, **5	4.023%
	all	*5, **4	*7 ,**6	0.6019%

Table 6.31: Hamming comparing α extreme limits

accuracy		$\alpha = 0.05$	$\alpha = 0.95$	difference
Chinese	w-H **	80.57%	**83.41%!**	2.835%
	w-FH **	78.3%	**82.97%!**	4.673%
German	w-H **	**82.31%!**	75.56%	-6.752%
	w-FH =	88.12%!	88.03%	-0.08547%
English	w-H **	**76.23%!**	72.55%	-3.678%
	w-FH =	77.3%	77.49%!	0.1875%
Catalan	w-H *	71.89%	72.71%!	0.8153%
	w-FH **	74.68%	**77.92%!**	3.235%
Spanish	w-H *	70.58%	71.71%!	1.129%
	w-FH **	73.4%	**76.55%!**	3.146%
Czech	w-H **	**71.02%!**	69.94%	-1.078%
	w-FH *	69.85%	70.3%!	0.4504%
Japanese	w-H **	**70.15%!**	61.21%	-8.94%
	w-FH **	**70.45%!**	61.88%	-8.57%
total	w-H	*4, **4	*3, **1	-2.238%
	w-FH	*1, **1	*4, **3	0.4337%
	all	*5, **5	*7 ,**4	-0.9023%

It is difficult to find out if lexical features are more important than syntactic ones or the other way around because it is very language dependent and frequently what is better for Hamming gives opposite results for Frame Hamming. Tables 6.30 and 6.31 compare the accuracy for extreme α values. Table 6.30 contrasts $\alpha = 0$ and $\alpha = 1$, and Table 6.31 contrasts $\alpha = 0.05$ and $\alpha = 0.95$.

In both tables lexical features ($\alpha \approx 0$) seem to perform better for Hamming and syntactic features ($\alpha \approx 1$) seem to perform better for Frame Hamming (except in Chinese). It is interesting to note that Frame Hamming already adds some lexical features gained by promoting samples of the same frame.

For Japanese data lexical features are always better than the syntactic ones, again, it may be the effect of a poor dependency relation annotation which is almost non-existent.

6.5 Contrasting different k for k-NN

The following figures show the accuracy of the system of a wanted[3] k value on the development data set. The x axis which is drawn in a logarithmic scale represents the k value and the y axis represent the accuracy.

[3]The system will peak the closest "k" value to the one suggested to fit the closest panel for an equivalent class.

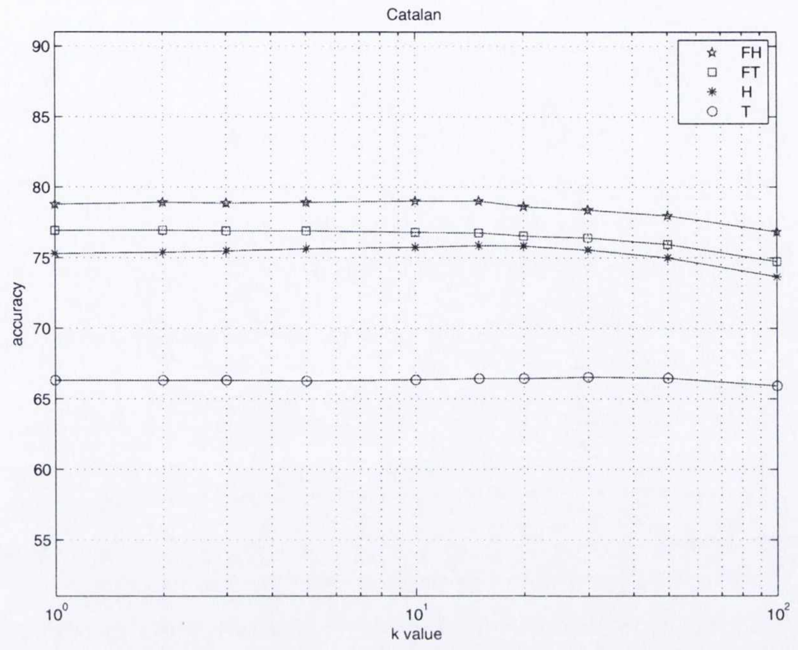


Figure 6.22: Tuning parameter k on Catalan development data set

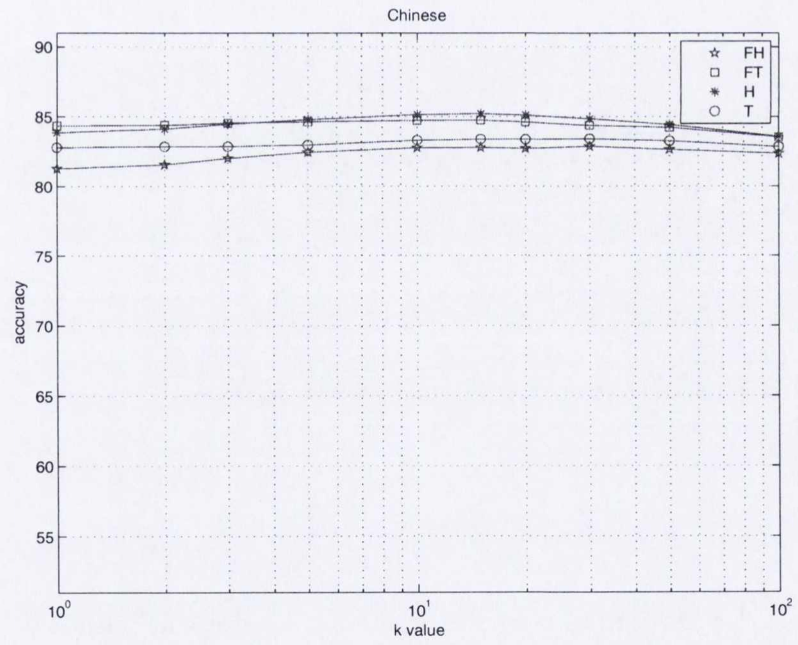


Figure 6.23: Tuning parameter k on Chinese development data set

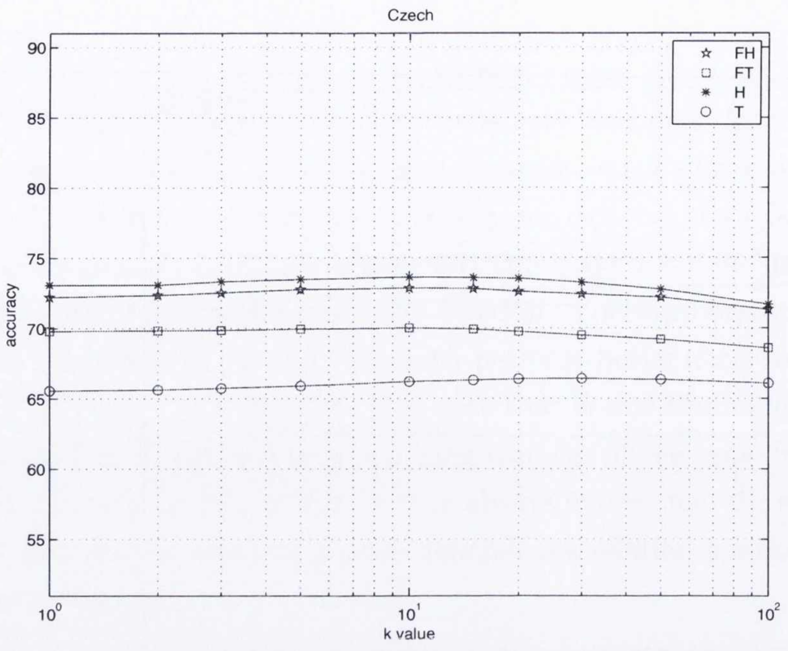


Figure 6.24: Tuning parameter k on Czech development data set

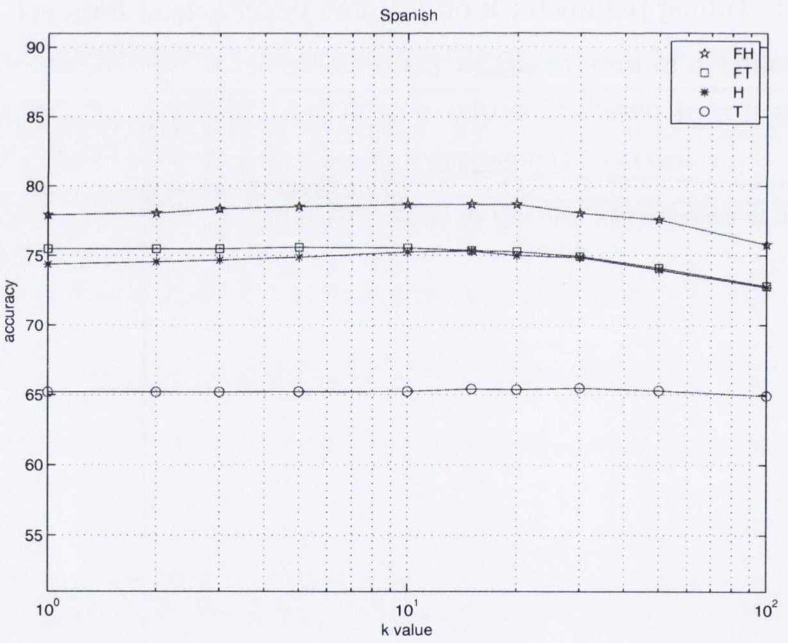


Figure 6.25: Tuning parameter k on Spanish development data set

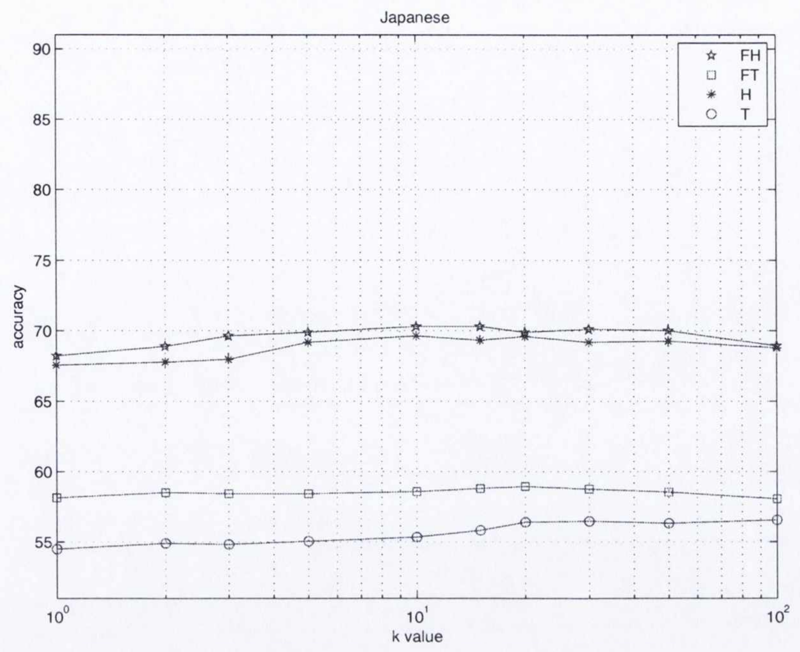


Figure 6.26: Tuning parameter k on Japanese development data set

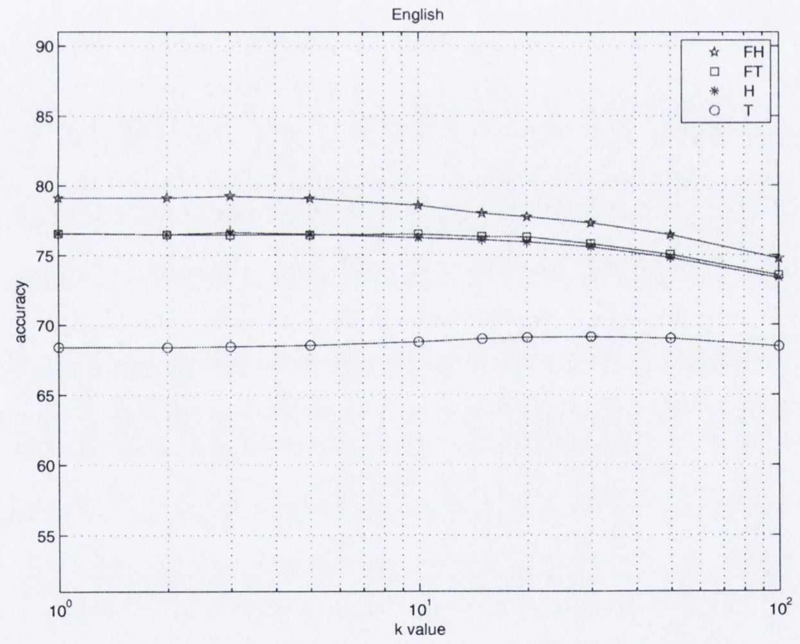


Figure 6.27: Tuning parameter k on English development data set

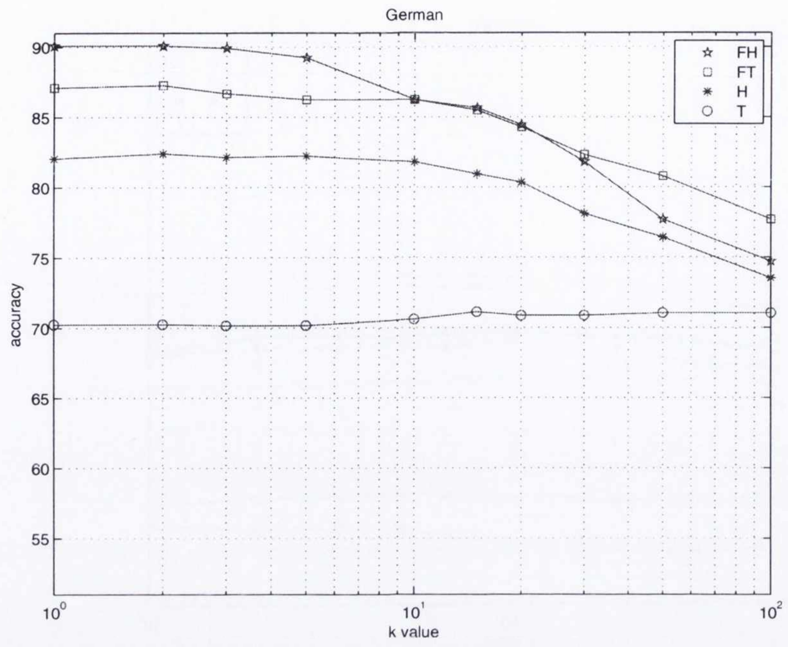


Figure 6.28: Tuning parameter k on German development data set

English and German seem to have its peak of performance at k=1 (for Frame Hamming), and as the k gets extended the accuracy gets degraded. A speculation about the huge difference between k=1 and k=10 of the German data set[4] can be based on the fact that the size of the training data set. German has a very small training data set in comparison with the other languages in number of sub-trees. Therefore, forcing the system to use more samples may cause the system to use less related samples which will decrease accuracy. For the other languages (Spanish, Catalan, Chinese and Czech) the graphs are very flat, swing and slightly increment of accuracy near k=10 but going down for higher values.

This observation is examined in Table 6.32, which contrasts the performance for k=1 with k=10. The results point that k=10 performs better than k=1 (except for German Frame Hamming), but the differences are marginal (less than 0.3% in average) in general and even smaller for the Frame versions (the ones with the higher performance).

Interestingly, with Frame Hamming the accuracy decreases faster than with Frame Ternary to the point that the accuracy lines cross each other.

[4]The differences are observable in Table 6.32 in which Frame Ternary and Frame Hamming show the largest drops in accuracy from k=1 to k=10.

Ternary has the most flat line because its first equivalence class is already very large, if the first equivalence class has more samples than the two k values that are compared the system will behave in an identical way for those two k values.

Ternary is the only atomic distance that seems to increase accuracy as the k value increases, it probably indicates that this measure is not adequate to select similar samples. Therefore, increasing the amount of samples the errors made by bad samples lose votes on the kNN system and the accuracy increases.

Table 6.32: Tree distance k=1 vs k=10

accuracy		k=1	k=10	difference
Chinese	T **	82.8%	**83.31%!**	0.5066%
	H **	83.88%	**85.14%!**	1.256%
	FT *	84.35%	84.73%!	0.3719%
	FH **	81.28%	**82.77%!**	1.493%
German	T =	70.26%	70.6%!	0.3419%
	H =	82.05%!	81.79%	-0.2564%
	FT =	87.09%!	86.24%	-0.8547%
	FH **	**90.09%!**	86.24%	-3.846%
English	T *	68.4%	68.77%!	0.3678%
	H =	76.58%!	76.24%	-0.3317%
	FT =	76.55%!	76.5%	-0.04327%
	FH *	79.11%!	78.57%	-0.5481%
Catalan	T =	66.31%	66.34%!	0.03469%
	H *	75.29%	75.73%!	0.4423%
	FT =	76.91%!	76.78%	-0.1301%
	FH =	78.77%	78.99%!	0.2255%
Spanish	T =	65.2%	65.24%!	0.03448%
	H **	74.37%	**75.24%!**	0.8706%
	FT =	75.48%	75.53%!	0.05172%
	FH *	77.92%	78.65%!	0.7327%
Czech	T **	65.56%	**66.2%!**	0.6419%
	H **	73.08%	**73.63%!**	0.5563%
	FT *	69.81%	70.02%!	0.2058%
	FH **	72.22%	**72.85%!**	0.6236%
Japanese	T *	54.49%	55.34%!	0.8496%
	H **	67.57%	**69.63%!**	2.069%
	FT =	58.15%	58.59%!	0.4433%
	FH **	68.23%	**70.3%!**	2.069%
total	T	*0, **0	*4, **2	0.3967%
	H	*0, **0	*5, **4	0.6579%
	FT	*0, **0	*2, **0	0.006376%
	FH	*2, **1	*4, **3	0.107%
	all	*2, **1	*15 ,**9	0.292%

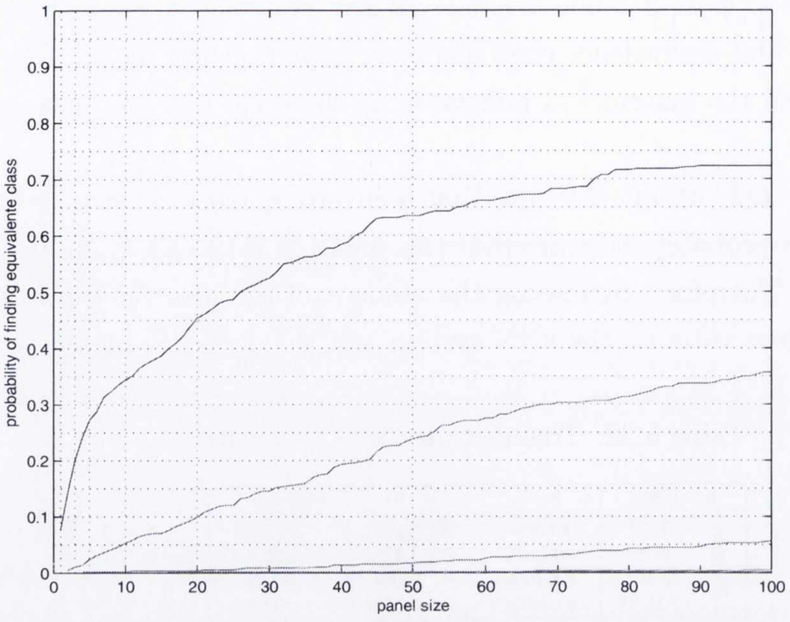


Figure 6.29: Equivalence class German Ternary
Probabilities of finding the n equivalence class under x amount of samples

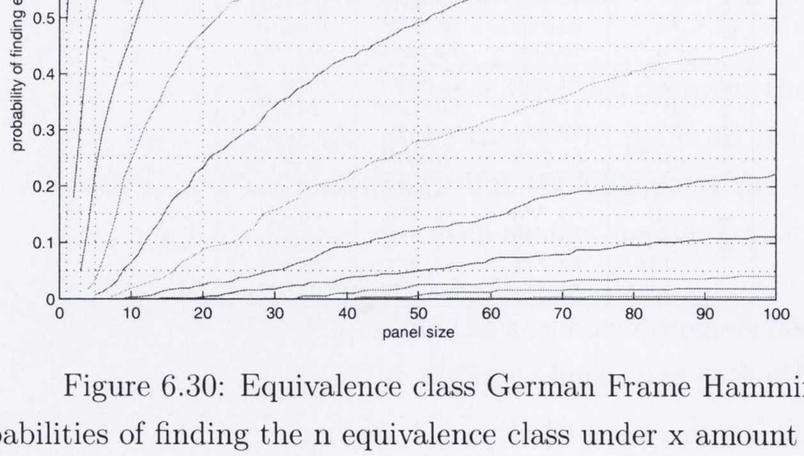


Figure 6.30: Equivalence class German Frame Hamming
Probabilities of finding the n equivalence class under x amount of samples

Figure 6.29 shows the probability of finding a particular equivalence class under certain amount of neighbours (or panel size) for Ternary system over the German development data set, for example the probability of finding the first equivalence class by looking at the first hundred neighbours is around 70%. Figure 6.30 shows the probability of finding a equivalence class under certain amount of neighbours for Frame Hamming system over German development data set, for example the probability of finding the first equivalence class by looking at the first ten neighbours is above 95%.

The differences on how easy it is to find the first and second equivalence class explains why there is so little variation on the accuracy of Ternary system: if the equivalence class is over the two values of 'k' that are being compared, the system will behave in a identical way for both values of k.

6.6 Frame Match cost

This section shows the effect of varying the frame mismatch cost on the development data set. The following figures show the accuracy of the system for different mismatching cost for each language. The x axis correspond to the mismatch cost and the y axis to the accuracy. It is intuitive to assume that negative mismatch costs degraded the accuracy of the system because they are to use samples from different semantic frame from the query that has to be labelled. This intuition is confirm in the following figures.

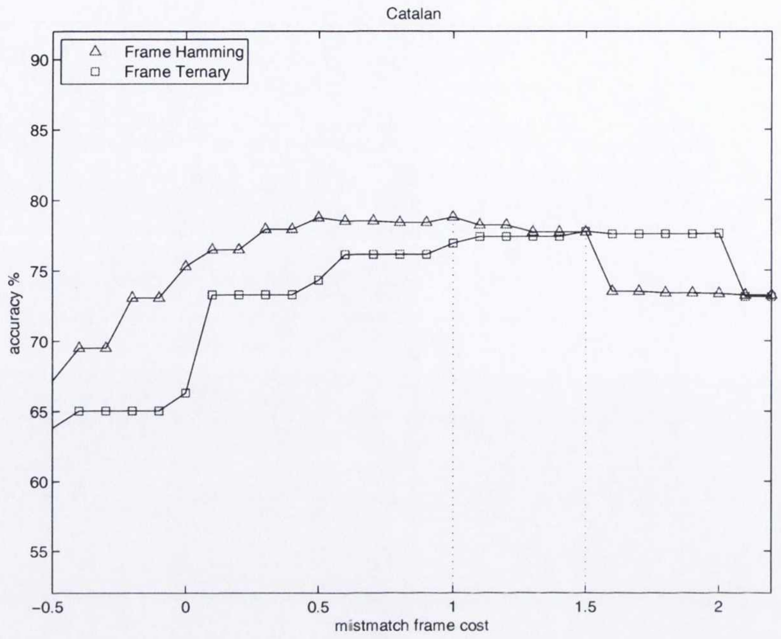


Figure 6.31: Tuning frame mismatch cost on Catalan development data

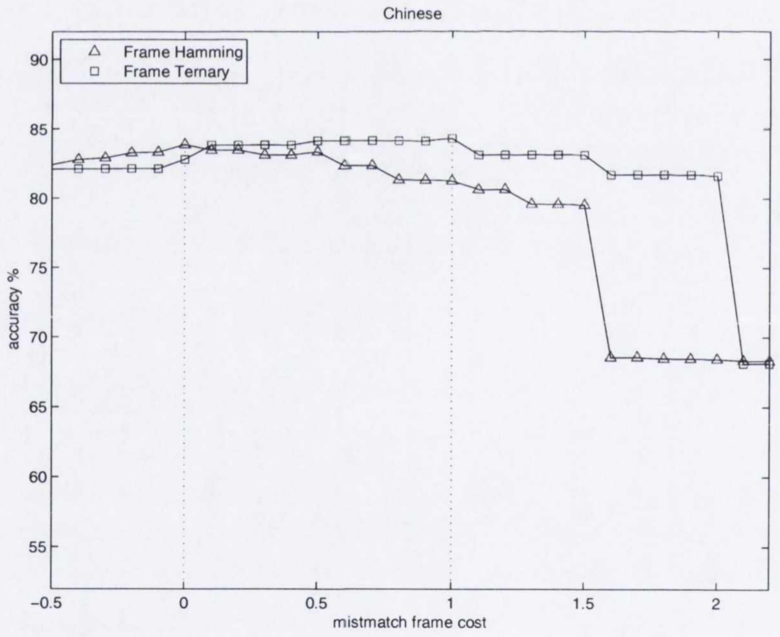


Figure 6.32: Tuning frame mismatch cost on Chinese development data

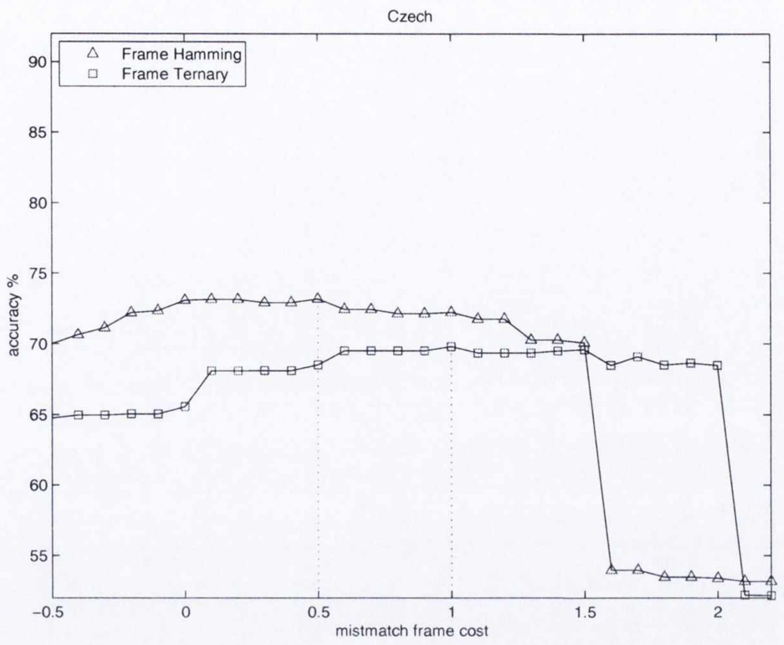


Figure 6.33: Tuning frame mismatch cost on Czech development data

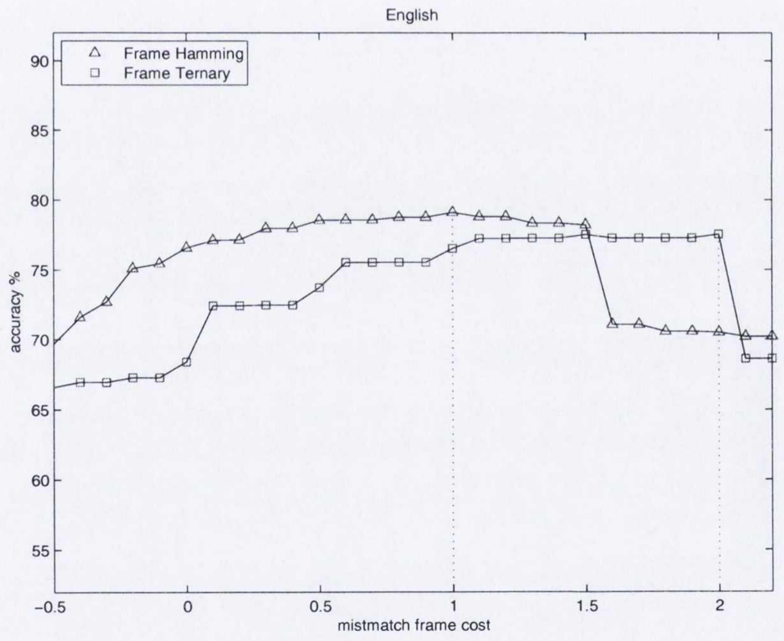


Figure 6.34: Tuning frame mismatch cost on English development data

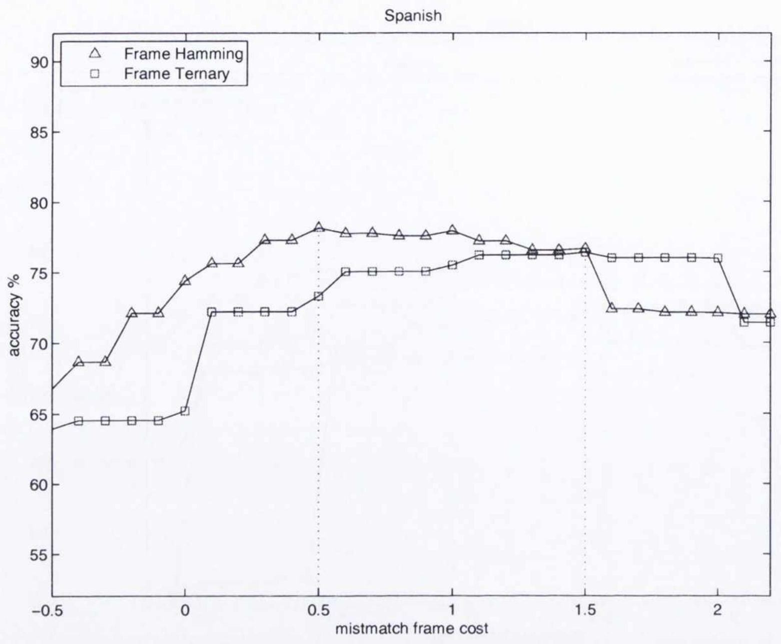


Figure 6.35: Tuning frame mismatch cost on Spanish development data

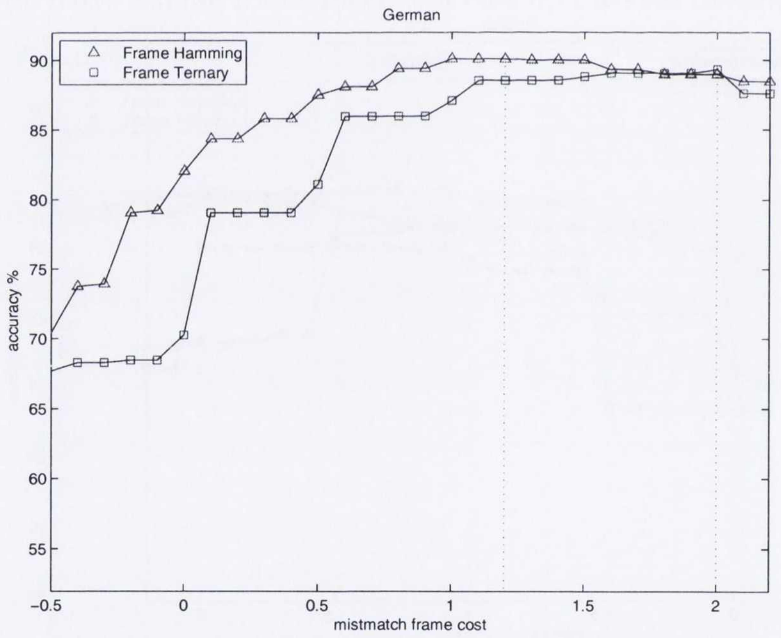


Figure 6.36: Tuning frame mismatch cost on German development data

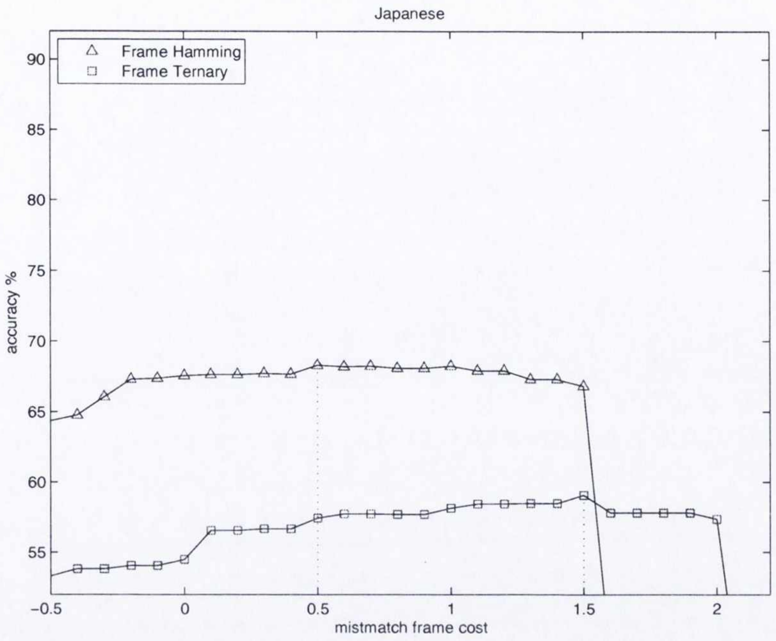


Figure 6.37: Tuning frame mismatch cost on Japanese development data

The results for optimizing the frame mismatch cost are very different from one language to another. Hence, some extra Tables are made to compare the trends in detail.

A notable observation is that Hamming drops its accuracy when the frame mismatch cost passes 1.5 (see details in Table 6.33), and Ternary when the frame mismatch cost pass 2 (see details in Table 6.33). Those two singularity points are remarkable for all languages except for German data in which the drop is very small in both cases and in one there is no significant difference, but still it is observable. This drop is due to the discard of samples from different frames to the sample to be labelled.

Table 6.33: Hamming frame cost 1.5 vs 1.6

accuracy		fm=1.5	fm=1.6	difference
Chinese	H **	**79.53%!**	68.56%	-10.97%
German	H =	**90%!**	89.32%	-0.6838%
English	H **	**78.23%!**	71.07%	-7.154%
Catalan	H **	**77.71%!**	73.5%	-4.215%
Spanish	H **	**76.64%!**	72.36%	-4.284%
Czech	H **	**70.08%!**	53.97%	-16.11%
Japanese	H **	**66.83%!**	46.99%	-19.84%
total	H	*6, **6	*0, **0	-9.036%

Table 6.34: Ternary frame cost 2 vs 2.1

accuracy		fm=2	fm=2.1	difference
Chinese	T **	**81.65%!**	68.13%	-13.52%
German	T **	**89.32%!**	87.61%	-1.709%
English	T **	**77.56%!**	68.61%	-8.943%
Catalan	T **	**77.6%!**	73.14%	-4.458%
Spanish	T **	**75.94%!**	71.42%	-4.525%
Czech	T **	**68.49%!**	52.21%	-16.29%
Japanese	T **	**57.37%!**	43.11%	-14.26%
total	T	*7, **7	*0, **0	-9.101%

In the case of Ternary, for a mismatch cost over 2, the cost of deleting and inserting both predicate nodes will be less than for swapping both of them even if the Dependency relation and POS labels are equal. In the case of Hamming, for a mismatch cost over 1.5, if predicates belong to different frames they are very likely to also differ word and lemma labels as well, which would add an extra cost of 0.5, what will make a cost over 2. What is more than inserting and deleting both predicate nodes.

In Hamming two samples can belong to different frames and still have the predicate nodes share the same word label, as the word can have a different meaning or frame, which explains the fluctuations for frame hamming for the mismatch range from 1.5 to 2. Hence, a mismatch cost over 1.5 will make most of the samples from other frames for Hamming unusable and a mismatch cost over 2 will make all samples from other frames for Ternary unusable. Over a mismatch cost of 2 the accuracy does not drop any longer, because all samples that belong to a different frame are unusable and increasing the cost of using them makes no difference. It can be observe that there is no difference between mismatch values of 2.1 and 2.2.

The Negative mismatch cost causes the system to avoid using samples from the same frame but still keep them usable. As expected, the accuracy drops as the mismatch cost becomes more negative. From what has been explained before it can be expected that optima mismatch cost should be in the range between 0 and 2.

Table 6.35: Dataset split by the maximum accuracy for different frame mismatch costs

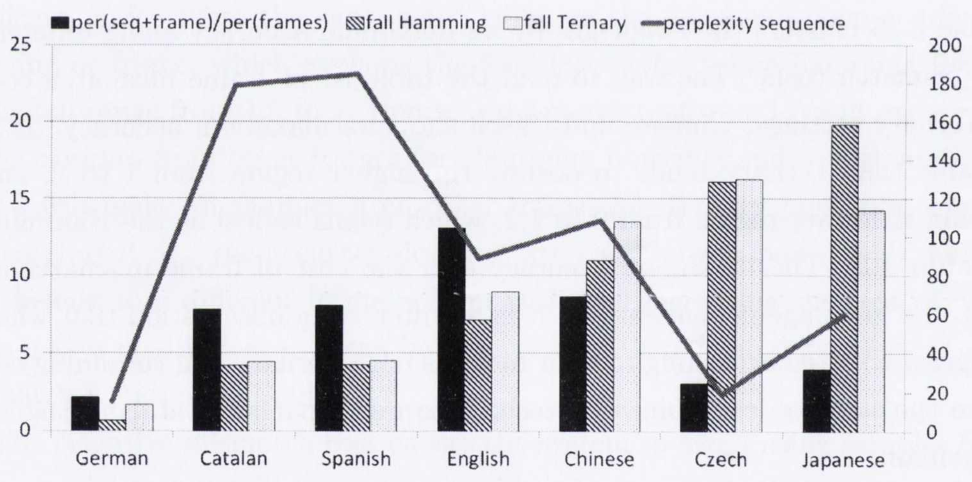
Max	Ternary	Hamming
0		Chinese
0.1		
0.2		
0.3		
0.4		
0.5		Czech , Japanese, Spanish*
0.6		
0.7		
0.8		
0.9		
1	Chinese, Czech	Catalan*,English
1.1		
1.2		German
0.3		
0.4		
1.5	Japanese, Spanish, Catalan	
1.6		
1.7		
1.8		
1.9		
2	English, German	

*Note: Spanish and Catalan show both a local maxima on hamming at 1 and at 1.5.

Table 6.35 clusters the languages by its maximum accuracy along different frame mismatch costs. The way to read the table is: at frame mismatch cost 1 for Ternary settings, Chinese and Czech show its maximum accuracy. It is observable that Ternary tends to occupy the higher region from 1 to 2, and Hamming the lower region from 0 to 1.2, which seems logical as the Hamming drops after 1.5. The maximum accuracy over the cost of frame mismatching seems to be language dependent and it is an interesting observation that when languages are sorted according to their maxims both Ternary and Hamming can produce the same order: Chinese, Czech, Japanese, Spanish, Catalan, English and German.

What makes German different? Probably the fact that German has fewer labels, makes the system to need less fewer training samples of each frame and consequently removing all samples of different frames will not lead to a great decrement on the accuracy of the system. In order to check this, the experiment described in Section 5.3.2 about calculating the perplexity of the chain of semantic arguments is replicated for all languages. Figure 6.38 illustrates the results.

- seq+frame: is the sequence of chains produced for each predicate created by concatenating all semantic labels in word order plus the label of the frame.
- frames: is the sequence of frame labels that appear on a data set.
- sequence: is the sequence of chains produced for each predicate created by concatenating all semantic labels in word order without the label of the frame.
- the function $per(x)$ calculates the perplexity of a sequence of items x.
- fall Hamming is the percentage of accuracy drop between Hamming frame mismatch cost 1.5 and 1.6.
- fall Ternary is the percentage of accuracy drop between Ternary frame mismatch cost 2 and 2.1.



	(normalized chain) per(seq+frame) /per(frames)	(brute chain) per(sequence)	% fall Hamming	% fall Ternary
German	2.18	15.11	0.68	1.71
Catalan	7.78	178.82	4.22	4.46
Spanish	8.08	184.89	4.28	4.53
English	13.04	88.45	7.15	8.94
Chinese	8.65	108.59	10.97	13.52
Czech	3.05	18.87	16.11	16.29
Japanese	4.02	59.99	19.84	14.26

Figure 6.38: Perplexity on sequences along languages

The column $per(seq + frame)/per(frames)$ (*normalized chain*), is an attempt to estimate how many sequences of roles per frame are. That is why the label of the frame is added to the sequence and later the perplexity is normalized by the perplexity of all frames. As it was expected German data set has the lowest normalized chain measure.

The column $per(sequence)$ (*brute chain*), is an attempt to estimate how many sequences of semantic roles are across frames. As the values were substantially larger than the other in the graph it is drawn using the secondary scale at the right side of the graph.

German language have the lowest fall, probably its because it have very small variety of semantic arguments and sequence of arguments (low *normalized chain*), what means it fewer samples of each frame to work well; for that reason removing all samples from other frames did not cause a big damage. Catalan and Spanish have a drop on accuracy, substantially larger than German, but relatively small in comparison with the other languages. (It can be observed that the *normalize chain* score is higher, which probably means it needs more samples to work well, but at the same time it also have a very high *brute chain* score, meaning that the samples from other frames may be very hard to use as they do not share the same sequences of roles, this second preposition may be the cause of having a moderate drop. Regarding Czech, this data set is substantially more annotated than any other language, it have the lowest ratio of arguments per predicate (see Figure 2.8 in page 27), at it contains many short sequences it is easy they repeat each other. Hence, the chain measures will be low miss-suggesting that the system needs a small amount of samples per frame to work well which is not the case).

Japanese is interesting because it the only language where Hamming falls more than Ternary, but this is a special case because for all other languages Ternary and Hamming almost fall from the same accuracy, but in the case of Japanese Ternary performs around 10% worse than Hamming at the dropping value.

6.7 Conclusion

The goal of the experiments of this chapter was to find optimal parameters for the task across all languages.

Regarding the variation of the deletion cost, for ternary, hamming and frame ternary most of the languages seems to show their maximum of accuracy at 0.5 and for frame hamming at 0.9. Distance settings with deletion cost equal to 0.5 will be explored in Chapter 7.

Regarding the weighting between POS and Dependency relation six of the seven language data set used show that equal weight leads to the maximum or very close to the maximum accuracy. Equal weighting is the default value adopted in the experiments of the previous chapter.

Regarding the weighting between lexical and syntactic features also for most of the languages converge on a maximum accuracy for equal weight. Equal weighting is the default value adopted in the experiments of the previous chapter.

Regarding the k value of the k-NN module, the experiments tends to show slightly higher accuracy at k=10 than at k=1. However the differences are negligible (less than 0.3 % in average).

Regarding the tuning of the frame mismatch cost, the experiments do not converge across different languages, in ternary the optimal value varies from 1 to 2 and in hamming from 0 to 1.2

The results of the experiments confirm that the default values of parameters are very close to the optimum.

Chapter 7

Contrasting Distance and Similarity

7.1 Introduction

This chapter compares tree-distance and tree-similarity. Section 7.1 exposes common assumptions on the equivalence of both concepts. Section 7.2 compares Tai-distance and Tai-similarity from a theoretical point of view defining, explaining and discussing three kinds of equivalences and showing that some of them hold while others do not. Most of its content has been already published by Emms and Franco-Penya (2012). And the final Section (7.3), experiment with Tai-distance and Tai-similarity in the Tree-SRL system showing the impact of the differences between both concepts.

Assumptions about equivalence of distance and similarity The comparison measures for arrays and trees are commonly defined as 'distances' or as 'similarities'.

Statements like the one shown in Figure 7.1 or the wikipedia entry on Figure 7.2, reflect a common belief: That 'distance' and 'similarity' notions are almost synonyms and that the measures are straightforwardly interchangeable notions, meaning that whatever can be produced by a 'distance' measure can be reproduced or dualized by a 'similarity' measure and vice-versa.

"To compare RNA structures, we need a score system, or alternatively a distance, which measures the similarity (or the difference) between the structures. These two versions of the problem -score and distance- are equivalent." (Herrbach et al., 2006)

Figure 7.1: Herrbach et al. (2006) statement about distance and similarities

Needleman and Wunsch formulated their problem in terms of maximizing similarity. "Another possibility is to minimize the edit distance between sequences, introduced by Vladimir Levenshtein." Peter H. Sellers showed (Sellers, 1974) that the two problems are equivalent.

Figure 7.2: Wikipedia entry on 'Needleman-Wunsh' (Wikipedia, 2012)

Such comments are common in the literature (Batagelj and Bren, 1995; Omhover et al., 2006; Lesot and Rifqi, 2010).

7.2 Distance vs Similarity: a theoretical study

It is convenient to define tree similarity in a similar way as tree distance was defined[1] but approaching the problem by maximizing a score assigned to an alignment, with swaps rewarded and deletions and insertions being punished.

Let C^{Θ} be a 'similarity' table, indexed by $\{\lambda\} \cup \Sigma$, where Σ is the alphabet of labels[2], and where $\alpha : S \mapsto T$ is any mapping from S to T, and then let $\Theta(\alpha : S \mapsto T)$ be defined by:

[1]See Section 3.3.2 at page 41 for details of tree distance definition.

[2]To keep notational overhead to a minimum, it will be use $\cdot$ for arbitrary members the label alphabet Σ.

Definition 7.1: ***Similarity scoring of an alignment***

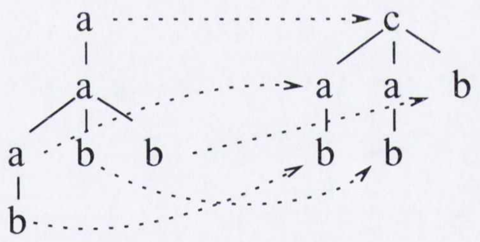
The similarity scoring of an alignment is the sum of the similarity score of all its swaps minus the similarity penalty for all inserted and deleted nodes:

$$\Theta(\alpha : S \mapsto T) = \sum_{(i,j)\in\mathcal{M}} C^{\Theta}(i^{\gamma}, j^{\gamma}) - \sum_{i\in\mathcal{D}} C^{\Theta}(i^{\gamma}, \lambda) - \sum_{j\in\mathcal{I}} C^{\Theta}(\lambda, j^{\gamma})$$

From this costing of alignments, a 'similarity' score on tree pairs is defined by maximisation:

Definition 7.2: ***Similarity scoring of a tree pair***

The Tree- or Tai-similarity $\Theta(S,T)$ between two trees S and T is the maximum *value of $\Theta(\alpha : S \mapsto T)$ over possible Tai-mappings from S to T, relative to a chosen cost table C^{Θ}*



with

$C^{\Theta}(x, \lambda) = C^{\Theta}(\lambda, x) = 0,$

$C^{\Theta}(x, x) = 2,\ C^{\Delta}(x, y) = 1$ *for* $x \neq y$,

the shown alignment score $\Theta(\alpha) = 9$

Copy of Figure 3.3 at page 45, similarity version.

Figure 7.3: An illustration of tree similarity.

Applied to the same example as shown in Figure 3.3 at page 45, which is duplicated as Figure 7.3 but with similarity score values, the shown alignment has score $\Theta(\alpha) = 9$, which is maximal for the given C^{Θ}.

$\Theta(S,T)$ can be computed via a simple modification of the (Zhang and

Shasha, 1989) algorithm[3].

Like for Δ, some settings of the C^{Θ}(cost-table) make little sense. Given the formulation in Equation 7.1, which subtracts the contribution from deletions and insertions, a setting where deletion or insertion cost entries are negative would give the counter-intuitive effect that a supertree of S would be more 'similar' (in the sense of higher Θ score) to S than S itself. For this reason it is nearly universally assumed that deletion and insertion entries in C^{Θ}can not be negative. Henceforth, it is assumed that 'similarity' Θ is always based on a table C^{Θ} that satisfies Equation 7.1.

(7.1) $\forall x, y \in \Sigma(C^{\Theta}(x, \lambda) \geq 0, C^{\Theta}(\lambda, y) \geq 0)$

7.2.1 Order-equivalence notions

For a given distance 'distance' Δ scoring[4] or a 'similarity' Θ scoring of alignments it can be defined at least the following three different kinds of orderings:

Definition 7.3: *Alignment ordering*

Given fixed S, and fixed T, rank the possible alignments $\alpha : S \mapsto T$ *by* $\Delta(\alpha : S \mapsto T)$

Definition 7.4: *Neighbour ordering*

Given fixed S, and varying candidate neighbours T_i, rank the neighbours T_i *by* $\Delta(S, T_i)$ *– typically used in k-NN classification.*

Definition 7.5: *Pair ordering*

Given varying S_i, and varying T_j, rank the pairings $\langle S_i, T_j \rangle$ *by* $\Delta(S_i, T_j)$ *– typically used in hierarchical clustering.*

The comparison these orderings motivates the following definition:

[3]On the domain of vertical trees this coincides with the well known approach to sequence comparison, the (alphabet-weighted) string similarity (Smith and Waterman, 1981; Gusfield, 1997).

[4]Δ first introduced on page 44.

Definition 7.6: ***A-dual***

C^Θ and C^Δ are A-dual if the alignment ordering they induce is reversed.

Definition 7.7: ***N-dual***

C^Θ and C^Δ are N-dual if the neighbour ordering they induce is reversed.

Definition 7.8: ***P-dual***

C^Θ and C^Δ are P-dual if the pair ordering they induce is reversed.

For example, the following examples 7.9 and 7.10 are A-duals in this sense. This will be proven in section 7.2.2.

Example 7.9: ***First A-dual example***

$$\Delta \textit{ with} \begin{cases} C^\Delta(x,\lambda) = 1 \\ C^\Delta(x,x) = 0 \\ C^\Delta(x,y) = 1 \end{cases} \quad \Theta \textit{ with} \begin{cases} C^\Theta(x,\lambda) = 0 \\ C^\Theta(x,x) = 2 \\ C^\Theta(x,y) = 1 \end{cases}$$

Example 7.10: ***Second A-dual example***

$$\Delta \textit{ with} \begin{cases} C^\Delta(x,\lambda) = 0.5 \\ C^\Delta(x,x) = 0 \\ C^\Delta(x,y) = 0.5 \end{cases} \quad \Theta \textit{ with} \begin{cases} C^\Theta(x,\lambda) = 0 \\ C^\Theta(x,x) = 1 \\ C^\Theta(x,y) = 0.5 \end{cases}$$

7.2.1.1 The conjectures

A natural question that presents itself is whether for every choice of C^Δ, if there is a choice of C^Θ which is an A-dual, N-dual or P-dual, and vice-versa. More precisely, there are the following conjectures:

$$\textbf{A-duality} \begin{cases} (i) & \forall C^\Delta \exists C^\Theta (C^\Delta \text{ and } C^\Theta \text{ are A-duals}) \\ (ii) & \forall C^\Theta \exists C^\Delta (C^\Delta \text{ and } C^\Theta \text{ are A-duals}) \end{cases}$$

$$\textbf{N-duality} \begin{cases} (i) & \forall C^\Delta \exists C^\Theta (\ C^\Delta \text{ and } C^\Theta \text{ are N-duals}) \\ (ii) & \forall C^\Theta \exists C^\Delta (\ C^\Delta \text{ and } C^\Theta \text{ are N-duals}) \end{cases}$$

$$\textbf{P-duality}\begin{cases} (i) & \forall C^{\Delta}\exists C^{\Theta}(\ C^{\Delta} \text{ and } C^{\Theta} \text{ are P-duals}) \\ (ii) & \forall C^{\Theta}\exists C^{\Delta}(\ C^{\Delta} \text{ and } C^{\Theta} \text{ are P-duals}) \end{cases}$$

It is important to know if these duality conjectures hold or not, because if they do not hold it would imply that there is a substantive difference with the outcomes achievable by distance and by similarities.

7.2.2 Alignment-duality results

The following lemma will be useful for considering the A-duality conjecture.

Lemma 7.11 *For any C^{Δ}, and some choice δ such that $0 \leq \delta/2 \leq min(C^{\Delta}(\cdot,\lambda), C^{\Delta}(\lambda,\cdot))$ let C^{Θ} be defined according to (i) below.*

For any C^{Θ}, and choice δ such that $0 \leq \delta \geq max(C^{\Theta}(\cdot,\cdot))$ let C^{Δ} be defined according to (ii) below.

$$(i) \begin{cases} C^{\Theta}(x,\lambda) = C^{\Delta}(x,\lambda) - \delta/2 \\ C^{\Theta}(\lambda,y) = C^{\Delta}(\lambda,y) - \delta/2 \\ C^{\Theta}(x,y) = \delta - C^{\Delta}(x,y) \end{cases} \quad (ii) \begin{cases} C^{\Delta}(x,\lambda) = C^{\Theta}(x,\lambda) + \delta/2 \\ C^{\Delta}(\lambda,y) = C^{\Theta}(\lambda,y) + \delta/2 \\ C^{\Delta}(x,y) = \delta - C^{\Theta}(x,y) \end{cases}$$

then in either case, for any $\alpha : S \mapsto T$

$$(7.2) \quad \Delta(\alpha) + \Theta(\alpha) = \delta/2 \times (\sum_{s\in S}(1) + \sum_{t\in T}(1))$$

Proof 7.12: ***Proof of Lemma 7.11***

If defining C^{Θ} from C^{Δ} by (i), by the choice of δ, the non-negativity restriction of $C^{\Theta}(x,\lambda)$ and $C^{\Theta}(\lambda,y)$ should be preserved. If defining C^{Δ} from C^{Θ} by (ii), by the choice of δ, the non-negativity restriction of all entries in C^{Δ} should be preserved.

Whether defining C^{Θ} from $C^{\Delta}$by (i), or C^{Δ} from $C^{\Theta}$by (ii), it is straightforward to show: $\Delta(\alpha) + \Theta(\alpha) = \delta/2 \times (2|\mathcal{M}| + |\mathcal{D}| + |\mathcal{I}|)$

But then Equation (7.2) follows since: $2|\mathcal{M}|+|\mathcal{D}|+|\mathcal{I}| = \sum_{s\in\mathcal{S}}(1)+\sum_{t\in\mathcal{T}}(1)$

Proof 7.13: ***Proof of alignment sum property***

$\Delta(\alpha) + \Theta(\alpha) = \delta/2 \times (2|\mathcal{M}| + |\mathcal{D}| + |\mathcal{I}|)$.

This is proven as follows:

If defining C^{Θ} from $C^{\Delta}$by (i), for $\Theta(\alpha)$:

$$\begin{aligned}
& \sum_{(i,j)\in\mathcal{M}} [\delta - C^{\Delta}(i,j)] - \sum_{i\in\mathcal{D}}[C^{\Delta}(i,\lambda) - \delta/2] - \sum_{j\in\mathcal{I}}[C^{\Delta}(\lambda,j) - \delta/2) \\
= \; & \delta(|\mathcal{M}| + \frac{|\mathcal{D}|}{2} + \frac{|\mathcal{I}|}{2}) - \sum_{(i,j)\in\mathcal{M}} [C^{\Delta}(i,j)] - \sum_{i\in\mathcal{D}}[C^{\Delta}(i,\lambda)] - \sum_{j\in\mathcal{I}}[C^{\Delta}(\lambda,j)] \\
= \; & \frac{\delta}{2}(2|\mathcal{M}| + |\mathcal{D}| + |\mathcal{I}|) - \Delta(\alpha)
\end{aligned}$$

If defining C^{Δ} from $C^{\Theta}$by (ii), for $\Delta(\alpha)$:

$$\begin{aligned}
& \sum_{(i,j)\in\mathcal{M}} [\delta - C^{\Theta}(i,j)] + \sum_{i\in\mathcal{D}}[C^{\Theta}(i,\lambda) + \delta/2] + \sum_{j\in\mathcal{I}}[C^{\Delta}(\lambda,j) + \delta/2) \\
= \; & \delta(|\mathcal{M}| + \frac{|\mathcal{D}|}{2} + \frac{|\mathcal{I}|}{2}) - \sum_{(i,j)\in\mathcal{M}} [C^{\Theta}(i,j)] + \sum_{i\in\mathcal{D}}[C^{\Theta}(i,\lambda)] + \sum_{j\in\mathcal{I}}[C^{\Delta}(\lambda,j)] \\
= \; & \frac{\delta}{2}(2|\mathcal{M}| + |\mathcal{D}| + |\mathcal{I}|) - \Theta(\alpha)
\end{aligned}$$

Hence in either case the claim holds.

Theorem 7.14 *A-duality (i) and (ii) hold*

Proof 7.15: ***Proof of Theorem 7.14***

A-duality (i): *define C^{Θ} according to (i) in Lemma 7.11. Given the constant summation property of Equation (7.2), the ordering on alignments by Δ must be the reverse of the ordering by Θ.*

A-duality (ii): *similarly define C^{Δ} according to (ii) in Lemma 7.11*

Example 7.16: ***Example 7.9 revisited***

The C^{Θ} of Example 7.9 at page 163 can be seen as derived from the C^{Δ} with $\delta = 2$. Table 7.1 shows outcomes for other choices of δ.

Table 7.1: Outcomes for different choices of δ

	C^{Δ}	$C^{\Theta}(\delta = 2)$	$C^{\Theta}(\delta = 1)$	$C^{\Theta}(\delta = 0)$
(x, λ)	1	0	0.5	1
(x, x)	0	2	1	0
(x, y)	1	1	0	-1

As a corollary one can obtain the following concerning how one similarity table can be 'shifted' to an equivalent one, and similarly for distance tables:

Corollary 7.17 ('shifting') *for any* ${C^\Theta}_1$*, an alignment equivalent* ${C^\Theta}_2$ *can be derived by the conversion (a) below, and for any* ${C^\Delta}_1$*, an alignment equivalent* ${C^\Delta}_2$ *can be derived by the conversion (b)*

$$(a)\begin{cases} {C^\Theta}_2(x,\lambda) = {C^\Theta}_1(x,\lambda) - \kappa/2 \\ {C^\Theta}_2(\lambda,y) = {C^\Theta}_1(\lambda,y) - \kappa/2 \\ {C^\Theta}_2(x,y) = {C^\Theta}_1(x,y) + \kappa \end{cases} \quad (b)\begin{cases} {C^\Delta}_2(x,\lambda) = {C^\Delta}_1(x,\lambda) + \kappa/2 \\ {C^\Delta}_2(\lambda,y) = {C^\Delta}_1(\lambda,y) + \kappa/2 \\ {C^\Delta}_2(x,y) = {C^\Delta}_1 + \kappa \end{cases}$$

Proof 7.18: *Proof of Corollorary 7.17*

(a) is the composition of (ii), for some δ_1, with (i), for some δ_2, giving $\kappa = \delta_2 - \delta_1$. (b) is the composition (i), for some δ_1, with (ii), for some δ_2, giving $\kappa = \delta_2 - \delta_1$

Example 7.19: ***Example 7.9 revisited again***

The three A-dualizing similarities $C^\Theta(\delta = 2)$, $C^\Theta(\delta = 1)$ and $C^\Theta(\delta = 0)$ derived from the unit-cost distance table using varying δ in the (i) conversion of Lemma 7.11 can be seen as related to each other by the (a) 'shifting' conversion of Lemma 7.17, with $\kappa = -1$ each time.

7.2.3 Neighbour and Pair ordering results

7.2.3.1 Distance to Similarity

The case of using $\delta = 0$ in the (i) conversion of Lemma 7.11 from C^Δ to C^Θ gives non-positive values for all non-deletion, non-insertion entries in C^Θ, and is an especially trivial case of dualizing a distance setting C^Δ, with the effect that $\Theta(S,T) = -1 \times \Delta(S,T)$. Because of this, this distance-to-similarity conversion not only makes A-duals, but also N-duals and P-duals.

Theorem 7.20 *N-duality (i) and P-duality(i) hold*

Proof 7.21: *Proof of Theorem 7.20*

By choosing $\delta = 0$ in the (i) conversion of Lemma 7.11 from C^{Δ} to C^{Θ}, it is obtained $\Theta(S,T) = -1 \times \Delta(S,T)$, and hence $\Theta(S_1,T_1) \leq \Theta(S_2,T_2) \Leftrightarrow \Delta(S_1,T_1) \geq \Delta(S_2,T_2)$

This distance-to-similarity by negation is well known. On the other hand, concerning similarity-to-distance, in the (ii) conversion of Lemma 7.11 from C^{Θ} to C^{Δ}, one can only choose $\delta = 0$ if all $C^{\Theta}(x,y) \leq 0$, and clearly there are many natural settings of C^{Θ} where that is not true, because there is a pair (x,y) for which $C^{\Theta}(x,y) > 0$ then in order to avoid negative distances δ should be positive, and this would change the neighbour ordering.

7.2.3.2 Similarity to Distance

The remaining order-equivalence conjectures of section 7.2.1 are *N-duality(ii)* and *P-duality(ii)*, concerning the similarity-to-distance direction. Of the remaining conjectures, *P-duality(ii)* is stronger than *N-duality(ii)*. It is fairly easily to see that *P-duality(ii)* does not hold:

Theorem 7.22 *P-duality (ii) does not hold: there are C^{Θ} such that there is no C^{Δ} such that C^{Θ} and C^{Δ} are P-duals.*

Proof 7.23: *Proof of Theorem 7.22*

It is clearly possible for C^{Θ} to be such that there is no maximum value for $\Theta(S,T)$. For instance for $C^{\Theta}(a,a) = C^{\Theta}(a,\lambda) = C^{\Theta}(\lambda,a) = 1$, its clear that $\Theta(a,a) = 1$, $\Theta(aa,aa) = 2$ and in general $\Theta(a^n,a^n) = n$.

Let C^{Θ} be any table defining a similarity with no maximum. On the other hand, for each C^{Δ} there will be minimum value of $\Delta(S,T)$.

Say a pair $< S,T >$ belong to a class $\{\Theta\}_x$ if $\Theta(S,T) = x$.
Say a pair $< S,T >$ belong to a class $\{\Delta\}_y$ if $\Delta(S,T) = y$.
Let be $[\Delta]_{d_0}$ the class that contain the set of pairs $< S,T >$ for which $\Delta(S,T)$ is the minimal.

Suppose some C^{Δ} is a P-dual to C^{Θ}. There should exist a bijection between the set of similarity classes $\{[\Theta]_s\}$ and the set of distances classes of $\{[\Delta]_d\}$. Some similarity class $[\Theta]_{s_v}$ of Θ must correspond to the minimum distance class $[\Delta]_{d_0}$.
Let $[\Theta]_{s_w}$ be a higher Θ class than $[\Theta]_{s_v}$.
Then $[\Theta]_{s_w}$ must correspond to some Δ class $[\Delta]_{d_z}$ distinct from $[\Delta]_{d_0}$, and since $[\Delta]_{d_0}$ is the distance-minimum, this must be a higher distance class. Then for $(S_0, T_0) \in [\Delta]_{d_0}$, and $(S_1, T_1) \in [\Delta]_{d_z}$ you have $\Delta(S_0, T_0) < \Delta(S_1, T_1)$, but also $\Theta(S_0, T_0) < \Theta(S_1, T_1)$. So the supposed dual C^{Δ} does not reverse the pair-ordering of C^{Θ}.

Example 7.24: ***Proof 7.23 illustrated***
Let $\Theta$and Δbe P-duals.

As distance and similarity are defined a sum of atomic costs (swaps, insertions and deletions), the distance and similarity between two empty trees have to be zero: $\Delta(-,-) = \Theta(-,-) =$ 0.

If $\Theta(a,b) > 0$ then $\Theta(a,b) > \Theta(-,-)$), what would imply that $\Delta(a,b) < 0 = \Delta(-,-)$. As negative distances are not permitted,$\Theta$and Δcan not be P-duals.

Of the order-relating conjectures of section 7.2.1 the only remaining one is *N-duality(ii)*[5]. It can be shown that there are neighbour-orderings by a Tai-similarity which cannot be dualized by any Tai-distance whose deletion and insertion costs are symmetric.

Theorem 7.25 *There is C^{Θ} such that there is no C^{Δ} with $C^{\Delta}(x, \lambda) = C^{\Delta}(\lambda, x)$ such that C^{Θ} and C^{Δ} are N-duals*

Proof 7.26: ***Proof of Theorem 7.25***

Let $S = aa$, and the set of neighbours be $\{a, aaa\}$.
Let $C^{\Theta}(a, a) = x > 0$, and $C^{\Theta}(a, \lambda) = C^{\Theta}(\lambda, a) = y > 0$.

[5]That is the question of whether every neighbour-ordering via some C^{Θ} can be replicated by a neighbour ordering via some C^{Δ}.

- For (aa, aaa), the alignments with 2,1, and 0 a-matches haves scores, $2x - y$, $x - 3y$ and $-5y$, respectively, so the alignments maximising Θ are those with two a-matches, and $\Theta(aa, aaa) = 2x - y$.

- For (aa, a), the alignments with 1 and 0 a-matches have scores $x - y$ and $-3y$, respectively, so the alignments maximising Θ have one a-match, and $\Theta(aa, a) = x - y$.

Consider what is required for the Θ-decreasing neigbour ordering to be: $[aaa,\ a]$,: $\Theta(aa, aaa) > \Theta(aa, a) \Leftrightarrow 2x - y > x - y$
So there is a Θ-decreasing neighbour-ordering $[aaa,\ a]$.

Let $C^{\Delta}(a, a) = x'$, and $C^{\Delta}(a, \lambda) = C^{\Delta}(\lambda, a) = y'$. Note this assumes symmetric insertion and deletion costs. For (aa, aaa), the alignments with 2,1, and 0 a-matches haves scores, $2x' + y'$, $x' + 3y'$ and $5y'$, respectively. Two cases can be distinguish: (i) $2y' < x'$ and (ii) $2y' \geq x'$.

- For case (i), $x' = 2y' + \epsilon$, for some no-zero $\epsilon > 0$, and the 2,1,and 0 a-matches scores become $5y' + 2\epsilon$, $5y' + \epsilon$ and $5y'$, respectively, so taking the minimum, $\Delta(aa, aaa) = 5y'$.

- For case (ii), $y' = x'/2 + \kappa$, for some $\kappa \geq 0$, and the 2,1,and 0 a-matches scores become $2.5x' + \kappa$, $2.5x' + 3\kappa$ and $2.5x' + 5\kappa$, respectively, and 2-match case is amongst the minimal cases, so $\Delta(aaa, aa) = 2.5x' + \kappa$.

For (aa, a), the alignments with 1 and 0 a-matches haves scores, $x' + y'$ and $3y'$ respectively. Again, two cases are distinguished: (i) $2y' < x'$ and (ii) $2y' \geq x'$.

- For case (i), the 1 and 0 a-matches scores become $3y' + \epsilon$ and $3y'$ respectively, so taking the minimum, $\Delta(aa, a) = 3y'$.

- For case (ii), the 1 and 0 a-match scores become $1.5x' + \kappa$ and $1.5x' + 3\kappa$ respectively, and the 1-match case is amongst the minimal cases, so $\Delta(aa, a) = 1.5x' + \kappa$.

Summarising the Δ possibilities:

	Δ(aa,aaa)	Δ(aa,a)
(i) 2y' < x'	5y'	3y'
(ii) 2y' $\geq$ x'	2.5x' + κ	1.5x' + κ

In neither case (i) nor case (ii) is it possible to achieve a Δ-ascending neighbour ordering $[aaa, a]$, which was the Θ-descending neighbour ordering which was achieved with the assumed C^{Θ}.

Remark *If the requirement about the N-dualizing C^{Δ} have $C^{\Delta}(x,\lambda) = C^{\Delta}(\lambda, x)$, is dropped then the argument does not go through. The Θ-descending neighbour ordering $[aaa, a]$ can be replicated by a Δ-ascending neighbour ordering with $C^{\Delta}(a,\lambda) > C^{\Delta}(\lambda, a)$. For most applications of alignment-based 'distances', such an asymmetric setting of deletion and insertion costs would be considered unnatural.*

7.2.3.3 Does N-duality imply P-duality?

This section attempts to find if N-duality implies P-duality and if that is the case it would mean that N-duality and P-duality are exactly the same duality.

<u>**Proof 7.27**</u>*: **<u>Partial pr</u>oof for N-duality implies P-duality***

If C^{Δ} and C^{Θ} are N-duals those imply C^{Δ} are C^{Θ} are P-duals?
$\Delta(S_0, T_0) < \Delta(S_1, T_1) \rightarrow \Theta(S_0, T_0) > \Theta(S_1, T_1)|$ *being $C^{\Delta}$and C^{Θ}P-duals ?*

Assuming that it is always possible to find a sequence n of trees $(X_1...X_n)$ between T_0 and S_1 such that would satisfy:
$\Delta(S_0, T_0) < \Delta(T_0, X_1) \leq \Delta(X_1, X_2) \leq ... \leq \Delta(X_n, S_1) \leq \Delta(S_1, T_1)$

As Θ is and Δ are P-duals and in the previous sequence of inequalities each item in the sequence is a distance comparison which shares one tree with each of its neighbour items It is possible to create a new sequence of inequalities by using the P-duality property: each $\Delta(X_{m-1}, X_m) < \Delta(X_m, X_{m+1})$ would infer a $\Theta(X_{m-1}, X_m) > \Theta(X_m, X_{m+1})$

what leads to:
$\Theta(S_0, T_0) > \Theta(T_0, X_1) \geq \Theta(X_1, X_2) \geq \cdots \geq \Theta(X_n, S_1) \geq \Theta(S_1, T_1)$

Therefore, as the proof can work in both directions:
$\Delta(S_0, T_0) < \Delta(S_1, T_1) \Leftrightarrow \Theta(S_0, T_0) > \Theta(S_1, T_1)$

Remark The completion of the proof is left as an open question. It only requires to demonstrate that it will be always possible to find the sequences of trees $S, X_0 ... X_n, T$, which is partly explained in the following partial proof:

Proof 7.28: ***Partial proof of existence of equisdistant trees sequence***

Let $\sum^{\#}$ *be a super alphabet which contains* $\sum$. *And Let* $C^{\Delta\#}$ *be a distance cost table that contains* C^{Δ}, *extending the entries by defining deletion, insertion at a cost C and swap costs for the new symbols of the alphabet in relation to the old symbols also at cost C.*

Then it is easy to see that it is always possible to find a sequence of $n = (2 * (m + p) + 1)$ *(excluding S and T)trees such that the distance from one to the next will be constant C, where m is the amount of nodes of the source tree (S) and p the amount of nodes of the target tree (T). This sequence of trees will pass though the empty tree (with no nodes).*

The difference between consecutive trees for the first m elements of the sequence, it that one node which content is a symbol $\in \sum$ *is swapped into a symbol* $\in \sum^{\#}$ *at a cost C.*

At position m of the sequence all nodes of that tree $(\in \sum^{\#})$. *The difference between consecutive trees between m and* $m * 2$, *is that any* X_{j+1} *have one node less than* X_j *and the deletion had a cost of C.*

From $m * 2$ *to* $m * 2 + p$ *the differences from one tree to the next would be adding a node of the* $\in \sum^{\#}$ *adopting the shape of T at cost C, and from* $m * 2 + n$ *to the last positions of the sequence the nodes, the nodes will be turning into the nodes of T also at a cost C.*

Henceforth, $if \Delta(a, a) = 0$ *then* ***it is possible to find a sequence of trees, such that consecutive trees are at equal distance of each*** *other.*

Remark It is left as an open question to find if it is possible to find a sequence of equidistant trees in case of having an atomic swapping cost between identical items over zero.

Example 7.29: ***Equidistance sequence of trees***
Let imagine an alphabet $\sum = \{a, b, c, d\}$, *an extended alphabet* $\sum^{\#} = \{a, b, c, d, a', b', c', d'\}$, *a tree* $S = [ab]$ *and a tree* $T = [cd]$. *possible sequence will be:*

$$S = X_0 = [ab], X_1 = [a'b],$$
$$X_2 = [a'b'],\ X_3 = [b'],$$
$$X_4 = [-],$$
$$X_5 = [c'],\ X_6 = [c'd'],$$
$$X_7 = [cd'],\ X_8 = [cd] = T.$$

7.2.4 Conclusions and Illustrations

In view of the outcomes noted in sections 7.2.2, 7.2.3.1 and 7.2.3.2 concerning the various ordering conjectures it can be concluded that:

- Any hierarchical clustering outcome achieved via Δ can be replicated via Θ, but not vice-versa.
- Any categorisation outcome using nearest-neighbours achieved via Δ can be replicated via Θ, but not vice-versa.

In this sense 'similarity' and 'distance' comparison measures on sequences and trees are not interchangeable.

As far as the author of this thesis is aware this aspect of the choice between a similarity-based versus a distance-based comparison measure on sequences or trees has not been noted before (Emms and Franco-Penya, 2012).

7.2.4.1 Comparison of neighbour orderings of A-duals

Lesot and Rifqi (2010) present an empirical quantified comparison to measure equivalence degrees on orderings (Kendall, 1945), which they used for information retrieval. An analogous study can quantify distance and similarity orderings on trees. Lemma 7.11 on page 165 (i) gives an A-dual conversion from distance to similarity. This conversion allows unlimited variants trough changing a parameter δ. This section analyses the degree of N-duality as δ is varied. Table 7.2 gives some distance and similarity settings: the first column gives the unit-cost settings for Δ and the columns to the right give different similarity settings C^{Θ} derivable by the (i) conversion of Lemma 7.11 as δ is varied through various values.

Table 7.2: Unit-cost distance setting and several A-dual similarity settings.

		dual C^{Θ} for varying δ						
	C^{Δ}	2	1.5	1	0.5	0.2	0.1	0
(x, λ)	1	0	0.25	0.5	0.75	0.9	0.95	1
(x, x)	0	2	1.5	1	0.5	0.2	0.1	0
(x, y)	1	1	0.5	0	-0.5	-0.8	-0.9	-1

Definition 7.30: ***Kendal-Tau with ties***

Let N^1 and N^2 be two assignments of ranks to the same set of objects, U (with the possibility of ties). Where:

$\mathcal{P}$ is the set of all two-element sets of distinct objects from U, n is the amount of objects in U, define a penalty function p on any $\{T_i, T_j\} \in \mathcal{P}$, such that

(i) $p(\{T_i, T_j\}) = 1$ *if the order in N^1 is the reverse of the order in N^2,*

(ii) $p(\{T_i, T_j\}) = 0.5$ *if there is a tie in N^1 but not in N^2 or vice-versa*

(iii) $p(\{T_i, T_j\}) = 0$ *otherwise.*

The Kendall-Tau distance (with ties) between N^1 and N^2, $\tau(N^1, N^2)$, is:

$$\tau(N^1, N^2) = \sum_{\{T_i,T_j\}\in\mathcal{P}} [p(\{T_i, T_j\})] \times \frac{2}{m \times (m-1)}$$

An experiment was conducted to quantify how close the similarities defined by the varying C^{Θ}tables come to being N-duals for the distance. Using a set of 1334 trees, full sentences in dependency tree structure from the English data set, repeatedly a tree S was chosen, and neighbour files $N_{\Delta}(S)$ and $N_{\Theta}(S)$ were computed, with $N_{\Delta}(S)$ the ordering of the remaining trees by ascending Δ, and $N_{\Theta}(S)$ the ordering by descending Θ. $N_{\Delta}(S)$ and $N_{\Theta}(S)$ were then compared by the Kendall-tau measure τ. For each δ the average of this τ comparison between the distance and similarity neighbour files is shown in Figure 7.4. Binary settings were used for Δ.

The bottom-left corner, for $\delta = 0$ is the special case of Lemma 7.11 which amounts to the well-known trivial distance-to-similarity conversion, $\Theta(S, T) = -1 \times \Delta(S, T)$, noted in section 7.2.3.1. In this case the distance and similarity neighbour files are identical. As the graph clearly shows, as δ increases, the neighbour files exhibit progressively greater difference in ordering, until at $\delta = 2$ the τ score is 0.73, which corresponds to a tendency more towards order reversal than to replication. This experiment confirms that although each of

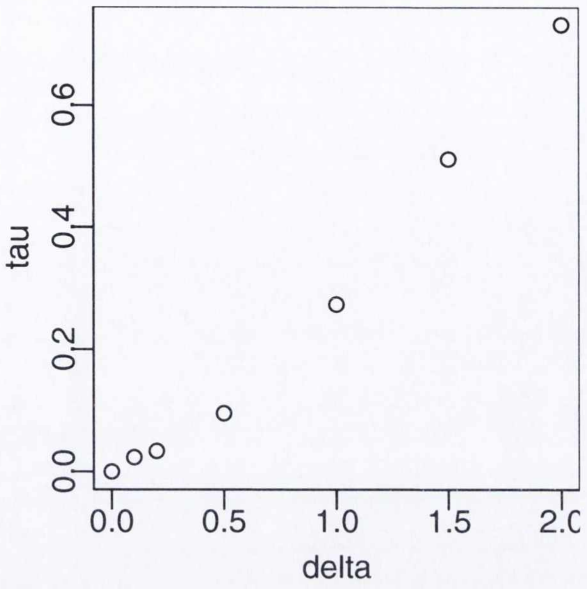


Figure 7.4: Average Kendall-tau comparison
Average Kendall-tau comparison on neighbours using distance and derived similarities. Distance setting is the first column of Table 7.2. Similarity settings are further columns of Table 7.2 defined by varying δ.

these similarity settings is an A-dual of the simple distance setting, they are not at all equivalent to each other as far as neighbour ordering is concerned.

The (ii) conversion of Lemma 7.11 on page 165 converts similarity settings to A-dual distance settings. Table 7.3 gives a similarity setting and then several distance settings derivable by the (ii) conversion as δ is varied through various values. Whilst the first experiment treated these simply as identical or not, for this second experiment, the base-line similarity node label is compared via $C^{\Theta}(x, y) = 1 - ham(x, y)$, where $ham(x, y)$ is the standard hamming distance explained in Section 4.7 at page 68.

Table 7.3: A similarity setting and several A-dual distance settings.

		dual C^{Δ} for varying δ						
	C^{Θ}	1	1.5	2	2.5	3	3.5	4
(x, λ)	0.5	1	1.25	1.5	1.75	2	2.25	2.5
(x, x)	1	0	0.5	1	1.5	2	2.5	3
(x, y)	0	1	1.5	2	2.5	3	3.5	4

Figure 7.5 plots the average τ comparison between the similarity and distance neighbour files, as δ is varied to give different distances. Again this experiment confirms that although each of the distance settings is an A-dual of the similarity setting, they are not equivalent to each other as far as neighbour ordering is concerned.

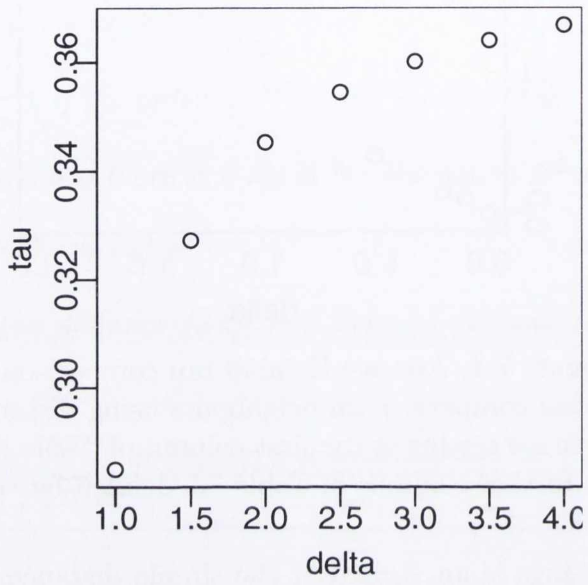


Figure 7.5: Average Kendall-tau comparison
Average Kendall-tau comparison on neighbours using a similarity and derived distances. Similarity setting is the first column of Table 7.3. Distance settings are further columns of Table 7.3 defined by varying δ.

7.2.4.2 Comparison of similarity and distance clusterings

Theorem 7.22 concerned the non-replicability by distance of pair-orderings by similarity. To illustrate this, consider a set of strings $\{a^5, a^4, a^3, a^2, a^1\}$. A table of pair-wise similarities of these was made with $C^{\Theta}(a,a) = 1, C^{\Theta}(a,\lambda) = 1$, and used to generate a single-link clustering,[6] shown as the the uppermost dendrogram in Figure 7.6.

No single-link clustering based on distance replicates this similarity clustering. The middle dendogram in Figure 7.6 is the result with $C^{\Delta}(a,a) = 0, C^{\Delta}(a,\lambda) = 1$, with all five shown on the same level because $\Delta(a^m, a^{m+1}) = 1$.

[6]So clusters are merged according to their 'best' single link.

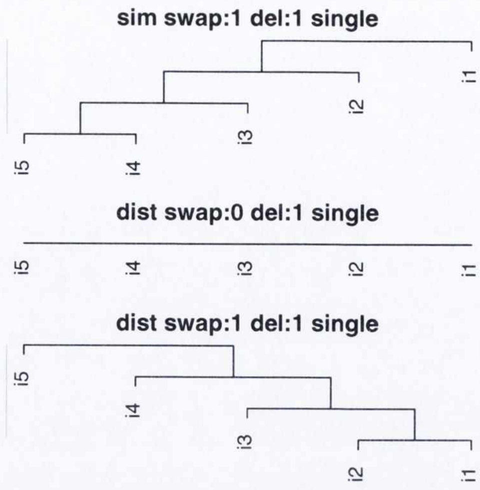


Figure 7.6: Similarity and distance clusterings.
The instance labels $i5 \dots i1$ represent $a^5 \dots a^1$.

The lowest dendogram in Figure 7.6 shows a result with $C^{\Delta}(a,a) = 1, C^{\Delta}(a,\lambda) = 1$. The same structure was found holding $C^{\Delta}(a,a) = 1$, and allowing the deletion/insertion cost to vary between 0.5 and 5.5 (which are $\geq C^{\Delta}(a,a)$) and between 0.4 and 0.1 (which are $< C^{\Delta}(a,a)$)

7.3 Distance vs. Similarity: compared in SRL

7.3.1 Contrasting Distance and Similarity

The first attempt to compare distance and similarity which will be explain in the following text. It was done by comparing tree edit distance with a A-dual similarity measure, a similarity measure which has deletion/insertion cost of **zero**, using (i) from Lemma 7.11, this means $\delta = 2$. Therefore, the swapping cost will range [1-2] because the swapping cost is defined as:

$$C^{\Theta}(x,y) = 2 - C^{\Delta}(x,y)$$

Table 7.4 shows the comparison. The differences between distance and similarity are very small but clear. The results indicate that tree distance performs better than similarity because it was never detected a strict significant difference in favour of similarity.

Table 7.4: Distance (d=1) vs similarity (swap[1-2])

accuracy		distance deletion=1	similarity swap[1-2]	difference
Chinese	T =	83%	83.04%!	0.03969%
	H *	83.93%	84.46%!	0.5304%
	FT =	84.78%	85.08%!	0.2995%
	FH *	81.79%	82.27%!	0.4835%
German	T **	**68.25%!**	65.08%	-3.166%
	H =	80.82%!	79.98%	-0.838%
	FT *	86.03%!	83.8%	-2.235%
	FH *	90.5%!	88.64%	-1.862%
o-German	T *	69.1%!	66.16%	-2.931%
	H =	72.53%!	72.03%	-0.5025%
	FT *	74.62%!	72.45%	-2.178%
	FH =	77.89%!	76.3%	-1.591%
English	T *	70.12%!	69.39%	-0.73%
	H *	77.57%!	77.04%	-0.5325%
	FT =	77.98%!	77.62%	-0.365%
	FH **	**79.86%!**	78.62%	-1.232%
o-English	T *	65.77%!	63.95%	-1.818%
	H =	66.68%!	65.94%	-0.7343%
	FT =	69.83%!	69.51%	-0.3147%
	FH =	69.69%!	69.09%	-0.5944%
Catalan	T **	**65.23%!**	63.44%	-1.783%
	H =	74.76%!	74.37%	-0.3902%
	FT =	76.77%	76.83%!	0.05321%
	FH *	78.51%!	77.66%	-0.8514%
Spanish	T **	**64.89%!**	63.53%	-1.353%
	H =	73.95%	74.55%!	0.6004%
	FT =	75.04%	75.58%!	0.5412%
	FH =	77.61%!	77.35%	-0.2537%
Czech	T **	**65.84%!**	65.2%	-0.6425%
	H **	**73.28%!**	72.58%	-0.6986%
	FT *	69.99%!	69.53%	-0.4615%
	FH **	**72.74%!**	70.65%	-2.083%
o-Czech	T **	**64.73%!**	63.86%	-0.8716%
	H **	**72.56%!**	71.03%	-1.534%
	FT **	**69.24%!**	68.17%	-1.066%
	FH **	**72.23%!**	69.32%	-2.903%
Japanese	T **	**56.69%!**	54.58%	-2.106%
	H **	**71.92%!**	68.59%	-3.329%
	FT **	**60.43%!**	58%	-2.426%
	FH **	**70.53%!**	66.77%	-3.762%
total	T	*9, **6	*0, **0	-1.536%
	H	*4, **3	*1, **0	-0.7428%
	FT	*5, **2	*0, **0	-0.8152%
	FH	*6, **4	*1, **0	-1.465%
	all	*24, **15	*2 ,**0	-1.14%

The only exception is the Chinese data set which accuracy is always better in tree similarity but only two times was a significant difference detected.

It looks odd that the swapping score of two nodes which are completely unrelated has to be more than zero. Therefore, a new set of experiments was designed in which tree similarity still has a cost of deletion/insertion equal to zero, but the swapping score ranges [0-1]. This is the A-dual of distance settings with the same swap cost as before but with deletion/insertion = 0.5 and so

$$C^{\Theta}(x, y) = 1 - C^{\Delta}(x, y)$$

Table 7.5 shows the comparison. The results are also very clear, for the Ternary versions works better with distance, but Hamming version works better with similarity, with the only exception of Japanese in simple Hamming, Japanese seem to be the only data set in which distance always performs better than similarity, this may happen because the sub-trees are far larger than in the other languages.

The hamming versions seems to work better for distance, when the costs are designed for distance, and for similarity, when the cost are designed for similarity. Therefore, in order to compare distance and similarity more fairly the maximum score obtained for the two sets of settings was used to make 7.6 table.

The results are in favour for tree distance, except for the case of Hamming which similarity performs slightly better, however in average tree distance seems to score 0.8% better.

The behaviour of the system over the Japanese data set is remarkably different from the rest. The results are always in favour of tree distance with a notable significant difference which came to be the largest average difference from all language data sets.

This is probably due to the fact that the sub-trees in the Japanese data set are larger than in any other data set, and tree similarity with its zero cost for deletions is giving preference to the largest sub-trees, which may not be optimal.

Figure 7.7 shows the alignment to the first neighbour of the second sub-tree extracted from the Japanese data set for Hamming similarity (Figure 7.7a) and for Hamming distance (Figure 7.7b). In Figure 7.7a, the empty nodes are

Table 7.5: Distance (d=0.5) vs similarity (swap[0-1])

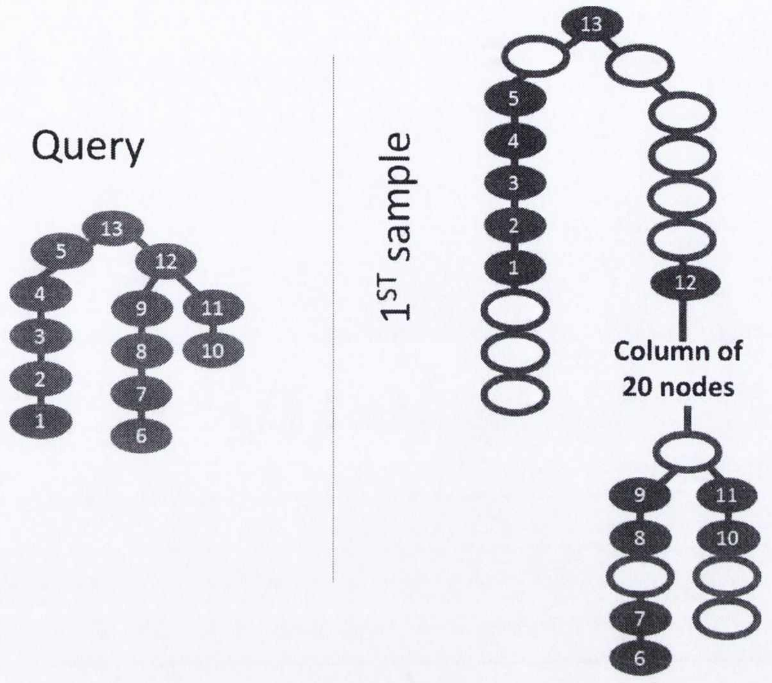
accuracy		distance deletion =0.5	similarity swap[0-1]	difference
Chinese	T =	83.98%!	83.86%	-0.1155%
	H **	85.06%	**85.88%!**	0.8227%
	FT *	85.88%!	85.53%	-0.3464%
	FH **	69.27%	**71.55%!**	2.284%
German	T =	67.41%!	65.74%	-1.676%
	H =	81.38%!	80.82%	-0.5587%
	FT **	**87.71%!**	85.01%	-2.7%
	FH =	88.36%	89.85%!	1.49%
o-German	T =	68.84%!	67.09%	-1.759%
	H =	72.7%	73.53%!	0.8375%
	FT *	75.46%!	73.03%	-2.429%
	FH **	53.6%	**57.37%!**	3.769%
English	T =	70.79%!	70.48%	-0.3092%
	H *	78.72%	79.43%!	0.7171%
	FT =	78.75%!	78.49%	-0.2619%
	FH **	70.05%	**73.27%!**	3.221%
o-English	T =	66.26%!	65.14%	-1.119%
	H **	64.9%	**67.52%!**	2.622%
	FT =	68.39%	69.34%!	0.9441%
	FH **	47.48%	**53.81%!**	6.329%
Catalan	T **	**65.7%!**	63.88%	-1.818%
	H =	75.8%!	75.58%	-0.2217%
	FT **	**79.35%!**	77.75%	-1.596%
	FH *	74.87%	75.56%!	0.6917%
Spanish	T **	**65.7%!**	63.98%	-1.717%
	H =	75.16%	75.64%!	0.4736%
	FT **	**77.25%!**	76.2%	-1.049%
	FH **	73.81%	**74.89%!**	1.082%
Czech	T **	**66.69%!**	66%	-0.6935%
	H **	74.05%	**74.77%!**	0.7138%
	FT **	**70.05%!**	69.37%	-0.6858%
	FH **	51.23%	**54.16%!**	2.924%
o-Czech	T =	65.08%!	64.73%	-0.3529%
	H **	72.23%	**73.36%!**	1.131%
	FT **	**68.19%!**	67.28%	-0.9076%
	FH **	51.48%	**53.37%!**	1.894%
Japanese	T **	**57.97%!**	55.86%	-2.106%
	H **	**75.34%!**	71.32%	-4.025%
	FT **	**61.52%!**	58.94%	-2.577%
	FH =	47.24%!	46.77%	-0.4702%
total	T	*4, **4	*0, **0	-1.167%
	H	*1, **1	*5, **4	0.2513%
	FT	*8, **6	*0, **0	-1.161%
	FH	*0, **0	*8, **7	2.321%
	all	*13, **11	*13 ,**11	0.06134%

Table 7.6: Max distance vs max similarity

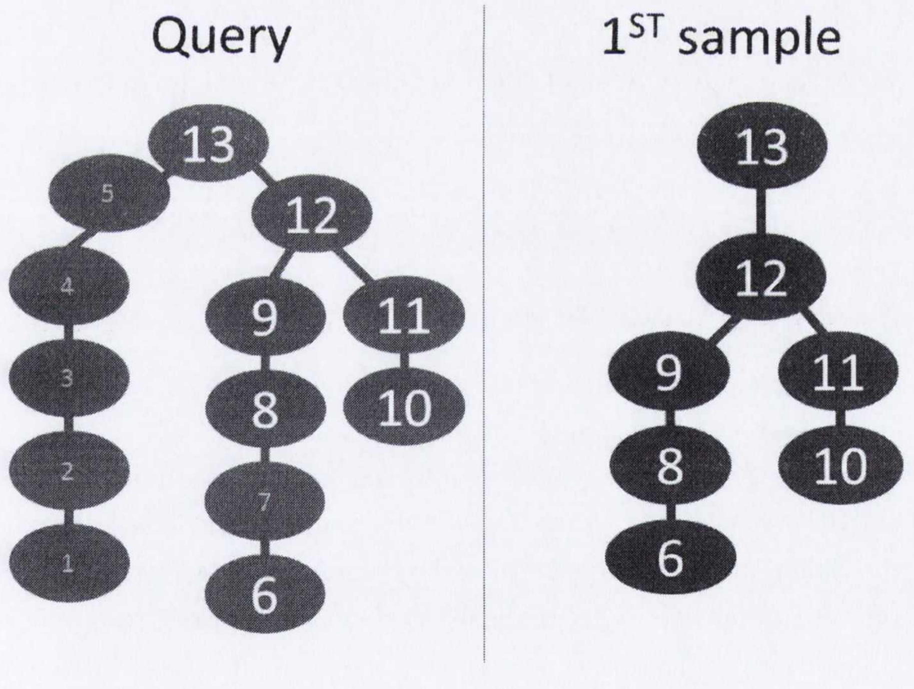
accuracy		max distance	max similarity	difference
Chinese	T =	83.98%!	83.86%	-0.1155%
	H **	85.06%	**85.88%!**	0.8227%
	FT *	85.88%!	85.53%	-0.3464%
	FH *	81.79%	82.27%!	0.4835%
German	T *	68.25%!	65.74%	-2.514%
	H =	81.38%!	80.82%	-0.5587%
	FT **	**87.71%!**	85.01%	-2.7%
	FH =	90.5%!	89.85%	-0.6518%
o-German	T *	69.1%!	67.09%	-2.01%
	H =	72.7%	73.53%!	0.8375%
	FT *	75.46%!	73.03%	-2.429%
	FH =	77.89%!	76.3%	-1.591%
English	T =	70.79%!	70.48%	-0.3092%
	H *	78.72%	79.43%!	0.7171%
	FT =	78.75%!	78.49%	-0.2619%
	FH **	**79.86%!**	78.62%	-1.232%
o-English	T =	66.26%!	65.14%	-1.119%
	H =	66.68%	67.52%!	0.8392%
	FT =	69.83%!	69.51%	-0.3147%
	FH =	69.69%!	69.09%	-0.5944%
Catalan	T **	**65.7%!**	63.88%	-1.818%
	H =	75.8%!	75.58%	-0.2217%
	FT **	**79.35%!**	77.75%	-1.596%
	FH *	78.51%!	77.66%	-0.8514%
Spanish	T **	**65.7%!**	63.98%	-1.717%
	H =	75.16%	75.64%!	0.4736%
	FT **	**77.25%!**	76.2%	-1.049%
	FH =	77.61%!	77.35%	-0.2537%
Czech	T **	**66.69%!**	66%	-0.6935%
	H **	74.05%	**74.77%!**	0.7138%
	FT *	70.05%!	69.53%	-0.5226%
	FH **	**72.74%!**	70.65%	-2.083%
o-Czech	T =	65.08%!	64.73%	-0.3529%
	H *	72.56%	73.36%!	0.7995%
	FT **	**69.24%!**	68.17%	-1.066%
	FH **	**72.23%!**	69.32%	-2.903%
Japanese	T **	**57.97%!**	55.86%	-2.106%
	H **	**75.34%!**	71.32%	-4.025%
	FT **	**61.52%!**	58.94%	-2.577%
	FH **	**70.53%!**	66.77%	-3.762%
total	T	*6, **4	*0, **0	-1.276%
	H	*1, **1	*4, **2	0.03983%
	FT	*8, **5	*0, **0	-1.286%
	FH	*5, **4	*1, **0	-1.344%
	all	*20, **14	*5 ,**2	-0.9664%

inserted, and in Figure 7.7b small grey numbers in the nodes of the query sub-tree represents deletions, in both sub-figures the matches are represented by sharing the same number across both trees, the numbers are given to the query in traversal post order. In the case of similarity this alignment represents the totality of the first equivalence class.

The query sub-tree (to be labelled) is the same for both alignments, only the training sub-tree changes. It can be seen that for this particular case similarity promotes a mappings to larger trees though making deletions/insertions free of cost. Large sub-trees like the one display in Figure 7.7a are seldom in other language data sets.



(a) Similarity



(b) Distance

Figure 7.7: Hamming alignments similarity and Distance, Japanese dataset Similarity settings tends to promote mappings to larger trees than distance settings in the ranking of neighbours, this is due to the zero cost for deletions and insertions.

7.3.2 Normalization effect

Theorem 7.31 *Normalising (dividing them by the amount of nodes) A-dual measures makes them N-duals and P-duals.*

<u>**Proof 7.32**: *Proof of theorem 7.31*</u>

As previously mentioned in Proof 7.12 on page 165 for any given mapping: Twice the amount of swaps plus all deletions and insertions sum exactly the amount of nodes in both trees together:

$$2|\mathcal{M}| + |\mathcal{D}| + |\mathcal{I}| = \sum_{s \in \mathcal{S}} (1) + \sum_{t \in \mathcal{T}} (1)$$

Therefore, normalising by amount of nodes in both trees is the same as normalizing by $2|\mathcal{M}| + |\mathcal{D}| + |\mathcal{I}|$ for any given mapping.
If equation 7.2 on page 165 is normalized, then:

$$\frac{\Delta(\alpha) + \Theta(\alpha)}{\sum_{s \in S}(1) + \sum_{t \in T}(1)} = \delta/2 \times \frac{\sum_{s \in S}(1) + \sum_{t \in T}(1)}{\sum_{s \in S}(1) + \sum_{t \in T}(1)}$$

That makes: $\hat{\Delta}(\alpha) + \hat{\Theta}(\alpha) = \delta/2$
As the value of δ is independent of the pair of trees for any mapping x:
$\hat{\Delta}(x) = \delta/2 - \hat{\Theta}(x)$
Let be α and β two mappings between different pairs of trees,
If $\hat{\Delta}(\alpha) < \hat{\Delta}(\beta)$ *then:*
$\delta/2 - \hat{\Theta}(\alpha) < \delta/2 - \hat{\Theta}(\beta)$
$\hat{\Theta}(\alpha) > \hat{\Theta}(\beta)$: *what makes them A-dual, N-dual and P-dual.*

The fact that normalised versions become A-dual and N-dual makes normalised tree distance with deletion = 1 performs in an identical way than normalised tree similarity with swapping cost [1-2] and normalised tree distance with deletion =0.5 performs in an identical way than normalised tree similarity with swapping cost [0-1]. This section compares the normalised measures to the brute measures.

7.3.2.1 Distance

The results clearly point that the brute measures perform better than the normalised ones, but even though 39 of 40 experiments show better accuracy on the brute measures and 26 times with strictly significant difference, the average difference is less than 1%. As they are both A-duals It is suspected that the brute measures rank better the neighbourhood.

The only case where the normalize version performs better than the brute version is Frame Ternary on the German data set. It happens that the differences between the brute and the normalize version on the prediction of the system are so seldom that the McNemar test fail to identify 25 differences, which is a requirement for the test to work well.

Table 7.7 compares tree distance with its normalize version for deletion cost equal to one.

Normalised distance promotes big trees because the cost is normalised by the amount of nodes they have. Large trees are likely to have more semantic relations than small ones what makes them more likely to contain infrequent relations and to predict them, and this can make the system to reduce the accuracy of the majority classes (or labels) and increase the accuracy of the minority classes.

It is observable in Table 7.8. It was decided that A0, A1 and A2 are the majority classes and A3 to A9 the minority classes ignoring any other class. Those classes are present in Chinese, German and English, in which all annotations are different but all use this set of labels.

As it was expected, for the majority classes, the brute measures performs in average better than the normalised measures, and for the minority labels, the normalised measures performs in average better than the brute measures. The small size of the samples of the minority classes creates the illusion of not being significantly different, when in reality what happen is that for every single comparison of the minority classes was not possible to find 25 different samples. Therefore, the McNemar test is not applicable.

Table 7.7: Tree edit distance vs normalized

		deletion cost = 1		
	accuracy	brute distance	normalize	difference
Chinese	T **	**83%!**	82.47%	-0.5377%
	H **	**83.93%!**	83.12%	-0.8047%
	FT **	**84.78%!**	84.34%	-0.4402%
	FH **	**81.79%!**	80.75%	-1.036%
German	T =	68.25%!	68.06%	-0.1862%
	H =	80.82%!	80.63%	-0.1862%
	FT =	86.03%	86.13%!	0.09311%
	FH =	90.5%!	90.41%	-0.09311%
o-German	T =	69.1%!	68.43%	-0.67%
	H =	72.53%!	71.69%	-0.8375%
	FT =	74.62%!	73.7%	-0.9213%
	FH **	**77.89%!**	76.05%	-1.843%
English	T **	**70.12%!**	69.52%	-0.6012%
	H **	**77.57%!**	76.5%	-1.074%
	FT **	**77.98%!**	77.44%	-0.5454%
	FH **	**79.86%!**	78.86%	-0.992%
o-English	T *	65.77%!	65.17%	-0.5944%
	H *	66.68%!	65.77%	-0.9091%
	FT *	69.83%!	69.06%	-0.7692%
	FH *	69.69%!	68.81%	-0.8741%
Catalan	T **	**65.23%!**	64.99%	-0.2394%
	H **	**74.76%!**	73.88%	-0.878%
	FT =	76.77%!	76.73%	-0.04434%
	FH **	**78.51%!**	77.54%	-0.9755%
Spanish	T *	64.89%!	64.68%	-0.2114%
	H **	**73.95%!**	73.33%	-0.6258%
	FT *	75.04%!	74.77%	-0.2706%
	FH **	**77.61%!**	76.83%	-0.778%
Czech	T **	**65.84%!**	65.25%	-0.5966%
	H **	**73.28%!**	71.99%	-1.293%
	FT **	**69.99%!**	69.36%	-0.6348%
	FH **	**72.74%!**	71.05%	-1.685%
o-Czech	T **	**64.73%!**	64.05%	-0.6843%
	H **	**72.56%!**	71.14%	-1.426%
	FT **	**69.24%!**	68.3%	-0.9364%
	FH **	**72.23%!**	70.61%	-1.613%
Japanese	T **	**56.69%!**	55.28%	-1.411%
	H **	**71.92%!**	69.74%	-2.182%
	FT **	**60.43%!**	58.98%	-1.448%
	FH **	**70.53%!**	68.18%	-2.351%
total	T	*8, **6	*0, **0	-0.5732%
	H	*8, **7	*0, **0	-1.022%
	FT	*7, **5	*0, **0	-0.5917%
	FH	*9, **8	*0, **0	-1.224%
	all	*32, **26	*0 ,**0	-0.8526%

Table 7.8: Distance vs normalized split by common and uncommon semantic relations

Deletion cost =1

	accuracy		Brute	Normalized	difference
Chinese	A0-2	T **	**87.44%!**	87.15%	-0.2871%
	A0-2	H **	**88.97%!**	88.49%	-0.4785%
	A0-2	FT *	90.72%!	90.51%	-0.2034%
	A0-2	FH **	**90.21%!**	89.29%	-0.9211%
	A3-9	T =	14.77%	15.91%!	1.136%
	A3-9	H =	36.36%	37.5%!	1.136%
	A3-9	FT =	37.5%	38.64%!	1.136%
	A3-9	FH =	55.68%	59.09%!	3.409%
German	A0-2	T =	74.46%!	74.25%	-0.2068%
	A0-2	H =	84.8%	84.8%!	0%
	A0-2	FT =	89.66%	89.66%!	0%
	A0-2	FH =	92.45%	92.45%!	0%
	A3-9	T =	12.15%	12.15%!	0%
	A3-9	H =	44.86%!	42.99%	-1.869%
	A3-9	FT =	53.27%	54.21%!	0.9346%
	A3-9	FH =	72.9%!	71.96%	-0.9346%
o-German	A0-2	T =	77.05%!	76.2%	-0.8467%
	A0-2	H =	79.68%!	78.65%	-1.035%
	A0-2	FT =	81.37%!	80.43%	-0.9407%
	A0-2	FH **	**83.91%!**	81.94%	-1.976%
	A3-9	T =	4.58%	5.344%!	0.7634%
	A3-9	H =	14.5%	15.27%!	0.7634%
	A3-9	FT =	19.85%!	19.08%	-0.7634%
	A3-9	FH =	29.01%!	28.24%	-0.7634%
English	A0-2	T **	**75.47%!**	74.88%	-0.59%
	A0-2	H **	**83.28%!**	82.27%	-1.016%
	A0-2	FT **	**84.58%!**	83.99%	-0.59%
	A0-2	FH **	**87.44%!**	86.23%	-1.209%
	A3-9	T =	28.55%!	28.24%	-0.312%
	A3-9	H =	50.08%!	49.61%	-0.468%
	A3-9	FT =	48.99%	49.61%!	0.624%
	A3-9	FH =	65.21%!	63.81%	-1.404%
o-English	A0-2	T *	74.34%!	73.54%	-0.8065%
	A0-2	H *	75.96%!	74.7%	-1.26%
	A0-2	FT **	**80.04%!**	79.08%	-0.9577%
	A0-2	FH **	**81.4%!**	79.79%	-1.613%
	A3-9	T =	5.714%	5.714%!	0%
	A3-9	H =	8.571%	8.571%!	0%
	A3-9	FT =	17.14%	17.14%!	0%
	A3-9	FH =	25.71%	25.71%!	0%
total	A0-2	T	*3, **2	*0, **0	-0.5474%
	A0-2	H	*3, **2	*0, **0	-0.758%
	A0-2	FT	*3, **2	*0, **0	-0.5383%
	A0-2	FH	*4, **4	*0, **0	-1.144%
	A3-9	T	*0, **0	*0, **0	0.3175%
	A3-9	H	*0, **0	*0, **0	-0.08749%
	A3-9	FT	*0, **0	*0, **0	0.3863%
	A3-9	FH	*0, **0	*0, **0	0.06142%
	A0-2	all	*13, **10	*0 ,**0	-0.7469%
	A3-9	all	*0, **0	*0 ,**0	0.1694%

Table 7.9: Tree similarity vs normalize

		swapping cost [0-1]		
accuracy		brute similarity	normalize	difference
Chinese	T **	**83.86%!**	83.26%	-0.599%
	H *	85.88%!	85.41%	-0.4691%
	FT =	85.53%!	85.29%	-0.2382%
	FH **	71.55%	**72.18%!**	0.6279%
German	T *	65.74%	67.88%!	2.142%
	H =	80.82%	81.19%!	0.3724%
	FT **	85.01%	**87.43%!**	2.421%
	FH =	89.85%	90.6%!	0.7449%
o-German	T *	67.09%	68.93%!	1.843%
	H =	73.53%!	72.95%	-0.5863%
	FT *	73.03%	75.13%!	2.094%
	FH =	57.37%	57.54%!	0.1675%
English	T =	70.48%!	70.35%	-0.1288%
	H =	79.43%!	79.29%	-0.146%
	FT =	78.49%	78.66%!	0.1718%
	FH **	73.27%	**74.17%!**	0.9018%
o-English	T =	65.14%	65.87%!	0.7343%
	H =	67.52%!	66.85%	-0.6643%
	FT =	69.34%	69.58%!	0.2448%
	FH =	53.81%	54.44%!	0.6294%
Catalan	T **	63.88%	**65.27%!**	1.392%
	H =	75.58%	75.65%!	0.07095%
	FT **	77.75%	**78.63%!**	0.878%
	FH *	75.56%	76.03%!	0.47%
Spanish	T **	63.98%	**65.07%!**	1.082%
	H *	75.64%!	74.93%	-0.7019%
	FT *	76.2%	76.89%!	0.685%
	FH =	74.89%	75.04%!	0.1438%
Czech	T =	66%!	65.92%	-0.07393%
	H =	74.77%	75.02%!	0.2549%
	FT *	69.37%	69.69%!	0.3238%
	FH **	54.16%	**55.55%!**	1.397%
o-Czech	T =	64.73%	64.74%!	0.01441%
	H *	73.36%	74.13%!	0.7635%
	FT =	67.28%	67.67%!	0.389%
	FH **	53.37%	**55.11%!**	1.736%
Japanese	T =	55.86%	56.7%!	0.8463%
	H **	71.32%	**74.57%!**	3.254%
	FT *	58.94%	60.33%!	1.392%
	FH **	46.77%	**49.67%!**	2.896%
total	T	*1, **1	*4, **2	0.7252%
	H	*2, **0	*2, **1	0.2148%
	FT	*0, **0	*6, **2	0.836%
	FH	*0, **0	*6, **5	0.9715%
	all	*3, **1	*18 ,**10	0.6869%

7.3.2.2 Similarity

Table 7.9 shows the results contrasting tree similarity with swapping cost [0-1] to its normalize version.

The results on tree similarity point that normalize similarity performs better than brute similarities, opposite results than with tree distance but same reason. Because tree similarity has a cost zero for deletion/insertion the system promotes very large trees, but this time normalizing will decrease the score of large trees because normalizing divides by the size of both trees. Again, big trees may contain infrequent semantic relations which would decrease the accuracy of the frequent labels and increase the accuracy of the infrequent ones, which would have a negative overall impact.

This is observable in Table 7.10. As in Table 7.8, only enumerated arguments were used for this test and only in English, German and Czech.

The set samples of the minority classes are still very small, but at least in all experiments concerning English in-domain data set it was possible to find 25 differences on the predictions, and in one case it was possible to detect a significant difference.

7.3.3 Combining normalized and brute?

It is difficult to combine normalized and brute measures because the scale of distances or similarities are very different, but what it is possible is to combine both measures by using brute measures to rank the neighbours and normalized measures to create the alignments or vice-versa. This experiment is out of the scope of the thesis.

7.4 Conclusion

This chapter introduced the concepts of Tai-distance and Tai-similarity on tree edit distance algorithms, and three order of equivalences: A-dual, if one can reproduce the same ranking of alignments for two given trees, N-dual if the neighbor ordering can be reproduced and P-dual if the pair ordering can be reproduced. Then it proves that the property of A-duality can be always hold,

Table 7.10: Similarity vs normalized split by common and uncommon semantic relations

			swap score[0-1]		
	accuracy		similarity	normalized	difference
Chinese	A0-2	T **	86.58%	**87.15%!**	0.5742%
	A0-2	H =	88.27%	88.49%!	0.2213%
	A0-2	FT *	90.02%	90.51%!	0.4905%
	A0-2	FH *	88.86%	89.29%!	0.4366%
	A3-9	T =	5.682%	15.91%!	10.23%
	A3-9	H =	46.59%!	37.5%	-9.091%
	A3-9	FT =	51.14%!	38.64%	-12.5%
	A3-9	FH =	65.91%!	59.09%	-6.818%
German	A0-2	T **	70.84%	**74.25%!**	3.413%
	A0-2	H =	83.76%	84.8%!	1.034%
	A0-2	FT **	86.97%	**89.66%!**	2.689%
	A0-2	FH *	91.11%	92.45%!	1.344%
	A3-9	T =	13.08%!	12.15%	-0.9346%
	A3-9	H =	45.79%!	42.99%	-2.804%
	A3-9	FT =	55.14%!	54.21%	-0.9346%
	A3-9	FH =	66.36%	71.96%!	5.607%
o-German	A0-2	T *	73.66%	76.2%!	2.54%
	A0-2	H =	78.65%	78.65%!	0%
	A0-2	FT =	79.21%	80.43%!	1.223%
	A0-2	FH =	81.66%	81.94%!	0.2822%
	A3-9	T =	5.344%	5.344%!	0%
	A3-9	H =	18.32%!	15.27%	-3.053%
	A3-9	FT =	17.56%	19.08%!	1.527%
	A3-9	FH =	32.82%!	28.24%	-4.58%
English	A0-2	T **	73.5%	**74.88%!**	1.379%
	A0-2	H **	81.44%	**82.27%!**	0.8236%
	A0-2	FT **	83.18%	**83.99%!**	0.8178%
	A0-2	FH **	85.05%	**86.23%!**	1.18%
	A3-9	T =	30.11%!	28.24%	-1.872%
	A3-9	H *	54.29%!	49.61%	-4.68%
	A3-9	FT =	51.17%!	49.61%	-1.56%
	A3-9	FH =	64.74%!	63.81%	-0.936%
o-English	A0-2	T **	70.16%	**73.54%!**	3.377%
	A0-2	H =	73.64%	74.7%!	1.058%
	A0-2	FT =	78.33%	79.08%!	0.756%
	A0-2	FH =	78.58%	79.79%!	1.21%
	A3-9	T =	11.43%!	5.714%	-5.714%
	A3-9	H =	11.43%!	8.571%	-2.857%
	A3-9	FT =	17.14%	17.14%!	0%
	A3-9	FH =	22.86%	25.71%!	2.857%
total	A0-2	T	*0, **0	*5, **4	2.256%
	A0-2	H	*0, **0	*1, **1	0.6275%
	A0-2	FT	*0, **0	*3, **2	1.195%
	A0-2	FH	*0, **0	*3, **1	0.8906%
	A3-9	T	*0, **0	*0, **0	0.3413%
	A3-9	H	*1, **0	*0, **0	-4.497%
	A3-9	FT	*0, **0	*0, **0	-2.694%
	A3-9	FH	*0, **0	*0, **0	-0.774%
	A0-2	all	*0, **0	*12 ,**8	1.242%
	A3-9	all	*1, **0	*0 ,**0	-1.906%

but not N and P-duality, everything that can be done by distance can be reproduce by similarity but not the other way around, the reason is because distance have a lower limit (negative distances are not distances) but similarity have no upper or lower limits. For the case in which distance and similarity are normalized, A-duality implies N- and P-duality. The second half of the chapter is an empirical comparison of the Tree-SRL system using distance or similarity, and their normalized versions.

It is difficult to obtain clear conclusions from the results, distance seem to perform better than similarity but this is probably due to the fact that in the case of similarity the similarity deletion/insertion score was fixed to zero, which promotes the influence of large sub-trees on the training data. The minority classes (A3 to A9) benefit from the promotion of large sub-trees but not the majority classes (A0-A2) for this reason promoting small sub-trees leads to higher accuracy.

Chapter 8

System variants

8.1 Introduction

This chapter examines alternative designs of the Tree-SRL system. It starts by comparing the results for two different methods for obtaining sub-trees. Then it compares two predictions methods: one using a different k-NN panel per argument and another using the same panel for all arguments. Then it contrasts the effect of the k-NN module against just picking random neighbours sorted by their distance, and finally as a final section it mentions stricter Tai-mapping for which experiments have not been carried out.

8.2 Sub-trees multiple vs one argument

This section examines the decision to include all arguments of each predicate in one sub-tree (one sub-tree per predicate) contrasting the accuracy of the system with the results obtained by including a single argument in the sub-tree (one sub-tree per argument).

The sub-trees with a single argument have fewer amount of nodes, therefore less context. For that reason it is expected that the accuracy of such a system would be lower than the one with multiple arguments, but the results suggest the opposite.

Observe the annotation for the predicate "expanding" on the following sentence:

Table 8.1: Multiple vs one arguments

accuracy		multiples arguments (large context)	single argument (small context)	difference
Chinese	T **	**83%!**	81.54%	-1.461%
	H **	83.93%	**87.95%!**	4.023%
	FT *	84.78%	85.31%!	0.5232%
	FH **	81.79%	**83.18%!**	1.389%
German	T **	**68.25%!**	62.94%	-5.307%
	H *	80.82%	83.99%!	3.166%
	FT =	86.03%	86.41%!	0.3724%
	FH =	90.5%	90.88%!	0.3724%
o-German	T *	69.1%!	66.92%	-2.178%
	H =	72.53%	73.53%!	1.005%
	FT =	74.62%!	74.2%	-0.4188%
	FH =	77.89%!	77.55%	-0.335%
English	T *	70.12%!	69.58%	-0.5454%
	H **	77.57%	**84.08%!**	6.506%
	FT **	77.98%	**80.61%!**	2.624%
	FH **	79.86%	**81.55%!**	1.696%
o-English	T =	65.77%	66.29%!	0.5245%
	H **	66.68%	**72.48%!**	5.804%
	FT **	69.83%	**72.76%!**	2.937%
	FH *	69.69%	71.89%!	2.203%
Catalan	T **	**65.23%!**	60.23%	-5.002%
	H **	74.76%	**79.77%!**	5.011%
	FT **	76.77%	**81.06%!**	4.283%
	FH **	78.51%	**81.44%!**	2.927%
Spanish	T **	**64.89%!**	59.92%	-4.964%
	H **	73.95%	**80%!**	6.047%
	FT **	75.04%	**79.17%!**	4.135%
	FH **	77.61%	**81.4%!**	3.789%
Czech	T **	65.84%	**66.65%!**	0.8082%
	H **	73.28%	**81.3%!**	8.023%
	FT **	69.99%	**73.78%!**	3.786%
	FH **	72.74%	**77.15%!**	4.418%
o-Czech	T =	64.73%	64.81%!	0.07203%
	H **	72.56%	**80.32%!**	7.758%
	FT **	69.24%	**72.8%!**	3.566%
	FH **	72.23%	**75.7%!**	3.472%
Japanese	T =	56.69%	57.74%!	1.053%
	H **	71.92%	**77.94%!**	6.018%
	FT **	60.43%	**63.08%!**	2.652%
	FH **	70.53%	**73.14%!**	2.614%
total	T	*6, **4	*1, **1	-1.7%
	H	*0, **0	*9, **8	5.336%
	FT	*0, **0	*8, **7	2.446%
	FH	*0, **0	*8, **7	2.255%
	all	*6, **4	*26 ,**23	2.084%

(39) A quarter of a century ago [national service]$_{A0}$ was promoted as a way of curing the manifest inequities of the draft – by, of all things, **expanding** [the draft]$_{A1}$.

represented in dependency tree structure in Figure 8.1.

From the predicate "expanding" it is possible to obtain a sub-tree with both arguments (Figure 8.2a) or two sub-trees with a single argument (Figure 8.2b). It is possible that the argument "service" could be well labelled without the need of the node "31 draft" in its sub-tree, and it is possible that the extra eight nodes over expanding give little help or even a negative impact for labelling the argument "draft".

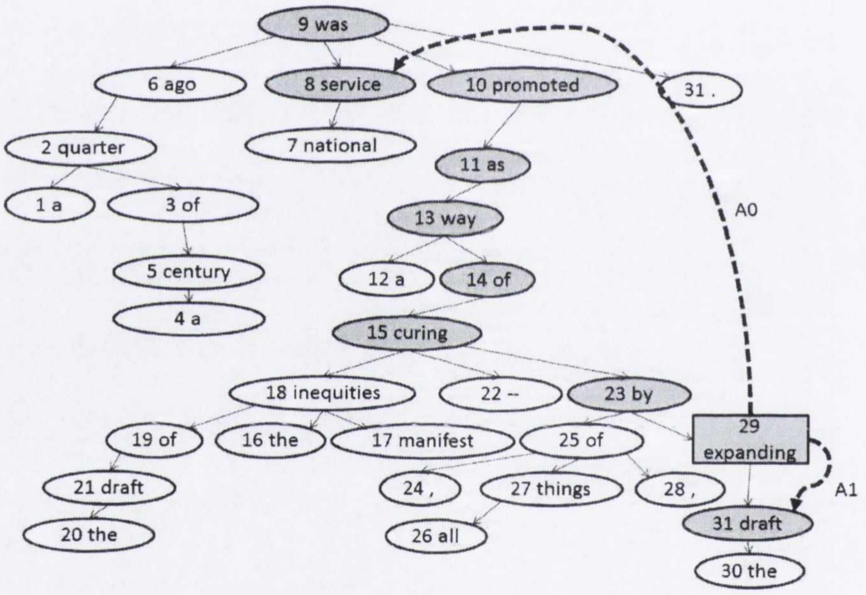
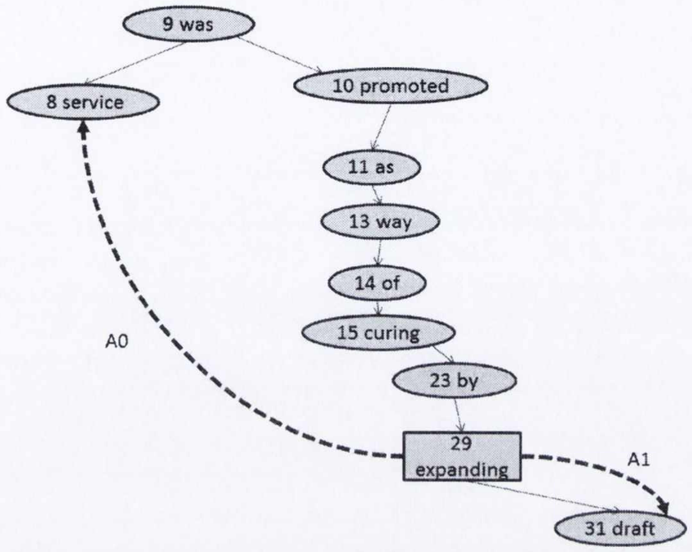


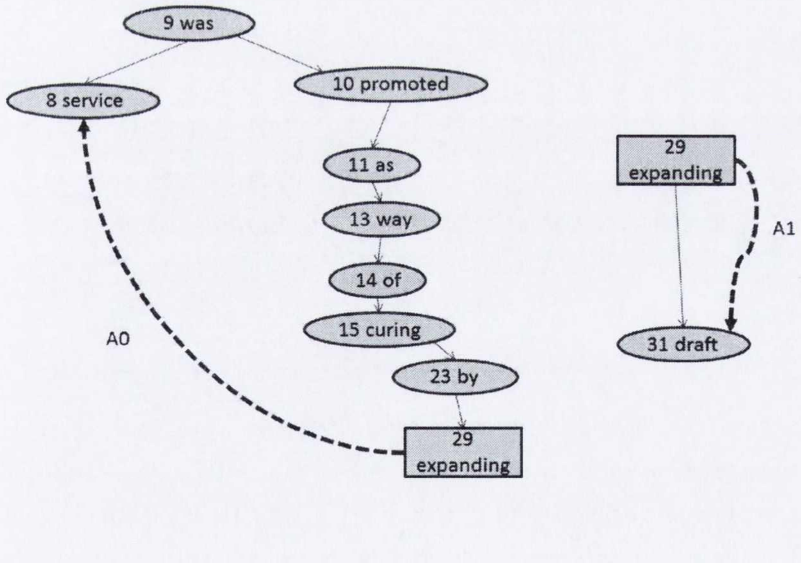
Figure 8.1: Sentence with long dependency

Table 8.2 shows the results for tree edit distance using sub-trees with multiple arguments and with a single argument. Except for Ternary system, all other measures indicate that one argument per sub-tree generates better accuracy.

This increment of accuracy is due to the increment on accuracy on the minority class labels (the less frequent ones). As it will be explained in the following sections, the majority class labels maintain their levels of accuracy, but



(a) multiple-arguments sub-tree



(b) single-arguments sub-tree

Figure 8.2: Equivalence classes on German Ternary
Probabilities of finding the nth equivalence class under x amount of samples. The uppermost line corresponds to the first equivalence class, lower lines for successively later equivalence classes.

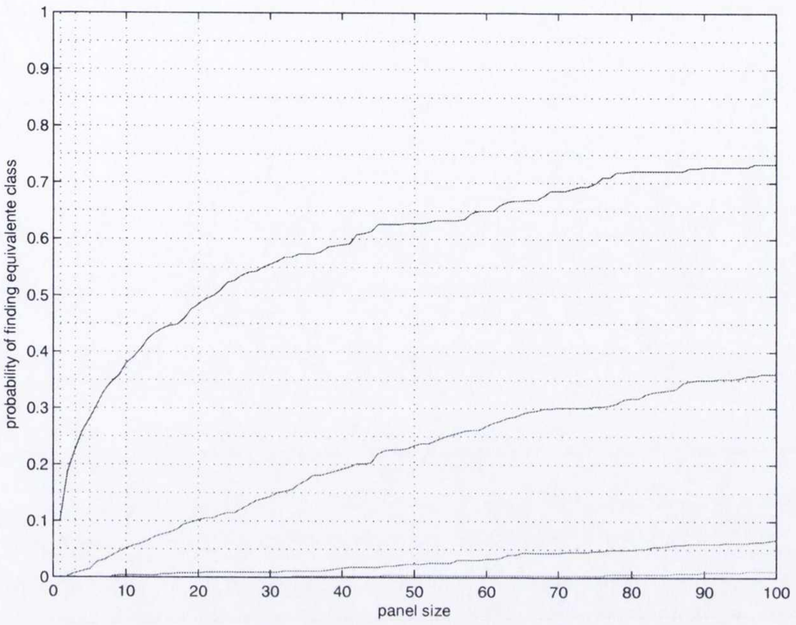
the minority ones are better labelled in one argument sub-tree. This probably means that the system is capable of finding a better panel for labelling those minority class labels if the context of the majority ones is removed.

8.2.1 Ternary

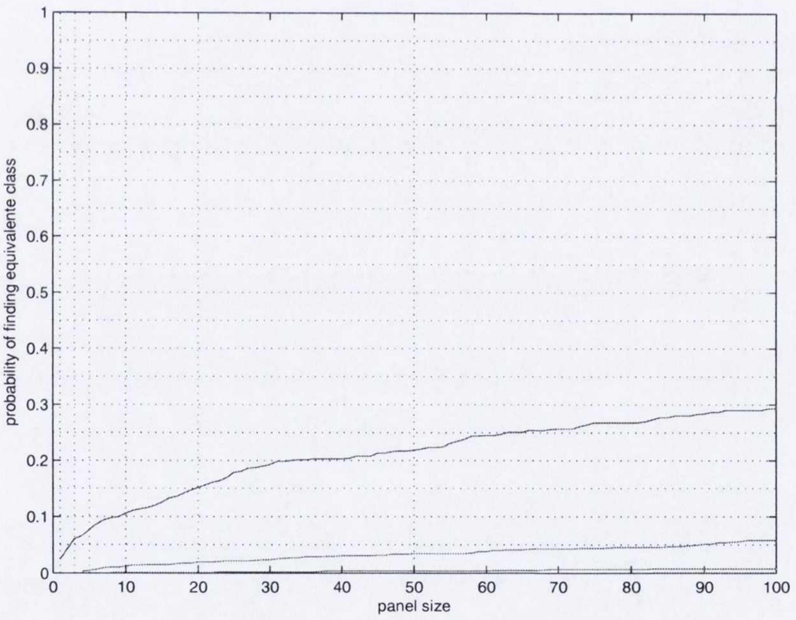
Figure 8.3a shows that when using multiple arguments per sub-tree the probability to find the first equivalence class on the first hundred samples is over 70% on Ternary settings, and Figure 8.3b shows that when using single argument per sub-tree, the probability of finding the first equivalence class on the first hundred samples is under 30% on Ternary settings. This extreme differences on the size of the first equivalence class was not observed in any other atomic cost setting (FT, H, FH).

Figure 8.4 shows the same comparison for Frame Ternary, in which the first three equivalence classes are easily observable. It is clear that the equivalence classes on the sub-trees of a single argument are slightly larger but the differences are not as notable as in the case of Ternary. This remark is important because Ternary seems to behave differently from all other atomic cost settings. Please note that the two graphs are not really comparable, because in the case of sub-trees containing one argument the amount of sub-trees extracted from the training data set is equal to the amount of arguments whereas in the case of multiple arguments the amount of sub-trees are equal to the amount of predicates. The amount of arguments is substantially higher than the amount of predicates. Therefore, the first hundred samples represent a different percentage of the training samples which makes both graphs are incomparable.

It seems that Ternary system using one argument per sub-tree includes too many samples in the first equivalence class for the system to perform well. Including too many samples is a symptom of a poor selection of samples. It may reduce the accuracy of the majority labels by adding noisy samples to the prediction panels, and it may reduce even more the accuracy of the minority labels because rare labels in a huge panel will have very little chance to win the voting. Table 8.2 shows the accuracy of the enumerated arguments in two groups; again A0 to A2 as the majority label groups, and A3 to A9 as the

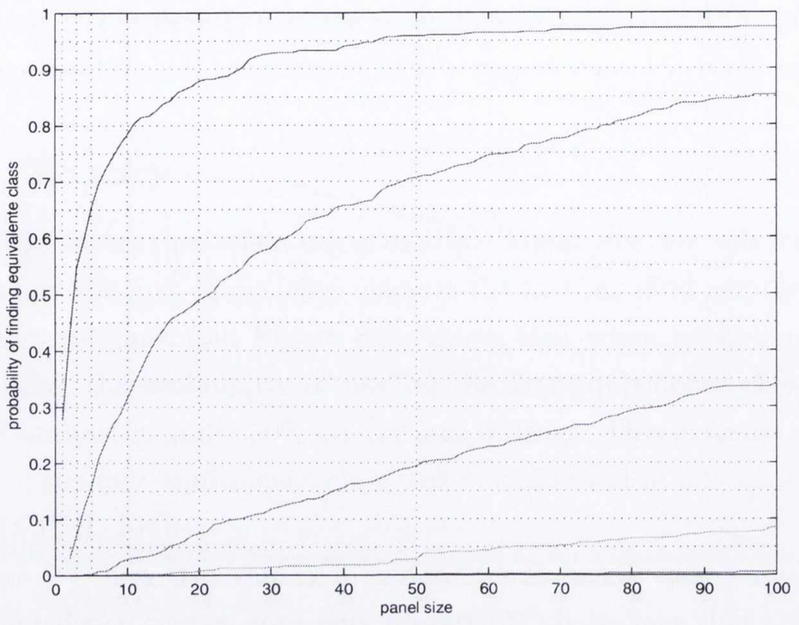


(a) multiple-arguments sub-tree

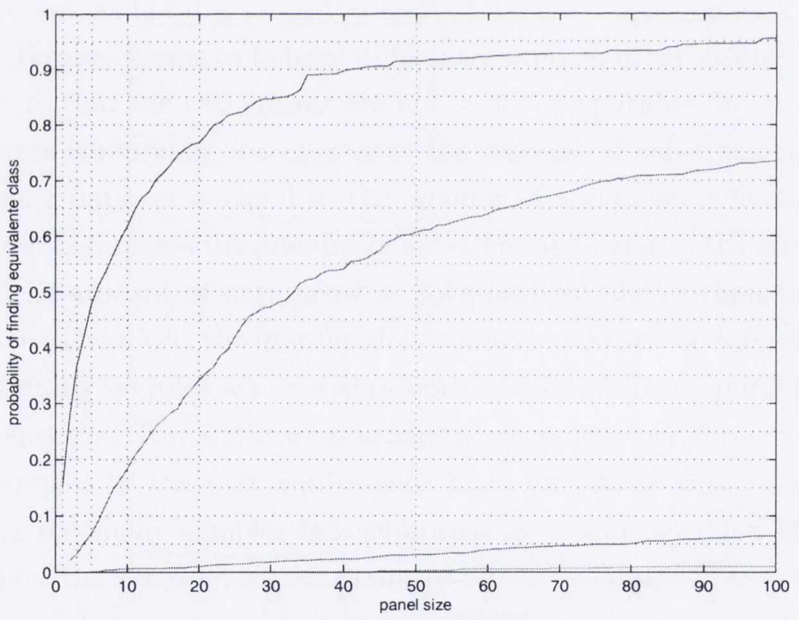


(b) single-arguments sub-tree

Figure 8.3: Equivalence classes on German Ternary
Probabilities of finding the nth equivalence class under x amount of samples. The uppermost line corresponds to the first equivalence class, lower lines for successively later equivalence classes.



(a) multiple-arguments sub-tree



(b) single-arguments sub-tree

Figure 8.4: Equivalence classes on German Frame Ternary
Probabilities of finding the nth equivalence class under x amount of samples.
The uppermost line corresponds to the first equivalence class, lower lines for
successively later equivalence classes.

minority labels group.[1]

As expected in Ternary with single-argument sub-tree settings reduce accuracy for majority and minority labels. This is substantially true for the case of the Chinese dataset on the minority labels, because the accuracy drops to zero.

Ternary settings may be too simple to perform well with one-argument sub-trees settings.

8.2.2 Frame Ternary, Hamming and Frame Hamming

As already said, the size of the equivalence classes did not change dramatically from multiple-argument sub-trees to single-argument sub-trees. It would be expected that adding extra samples in any case would lead to an increment of the most common labels on the training data set because they are more frequent on the samples: an increment of accuracy of the majority labels, and would also decrement the accuracy of minority labels. The decrement on minority labels is expected to be due to the fact that rare labels would have less chances to win the voting. However, results point the opposite. Majority labels sightly reduce accuracy (0.64% on average), and minority labels greatly increase accuracy with an important difference of 14.36% on average.

8.2.2.1 Discussion

Probably less context makes it easy to identify relevant samples for a given un-labelled semantic relationship. This may be related to the fact that all deletions, insertions or swaps have the same weight on the alignment regardless of how near they are from the two nodes involve in the un-labelled semantic relationship that have to be predicted.

It seems that one argument per sub-tree substantially benefits minority classes, the cause may be that there are few useful samples with the correct labels available in the firsts equivalent classes in the case of multiple argument sub-trees. Therefore less context may make those useful samples easy to be identified and it promotes them in the neighbourhood list.

[1]Majority and minority classes were defined for the first time in section 7.3.2.1 on page 187.

Table 8.2: Multiple vs one argument split into majority and minority labels

accuracy			multiples arguments	single argument	difference
Chinese	A0-2	T **	**87.44%!**	84.14%	-3.302%
	A0-2	H **	88.97%	**90.45%!**	1.483%
	A0-2	FT =	90.72%	90.97%!	0.2572%
	A0-2	FH *	90.21%!	89.63%	-0.5802%
	A3-9	T =	14.77%!	0%	-14.77%
	A3-9	H **	36.36%	**68.18%!**	31.82%
	A3-9	FT **	37.5%	**75%!**	37.5%
	A3-9	FH =	55.68%	84.09%!	28.41%
German	A0-2	T **	**74.46%!**	69.7%	-4.757%
	A0-2	H *	84.8%	87.59%!	2.792%
	A0-2	FT =	89.66%!	89.56%	-0.1034%
	A0-2	FH =	92.45%	92.86%!	0.4137%
	A3-9	T =	12.15%!	1.869%	-10.28%
	A3-9	H =	44.86%	51.4%!	6.542%
	A3-9	FT =	53.27%	57.94%!	4.673%
	A3-9	FH =	72.9%	72.9%!	0%
o-German	A0-2	T =	77.05%!	74.88%	-2.164%
	A0-2	H =	79.68%	79.87%!	0.1881%
	A0-2	FT =	81.37%!	79.77%	-1.599%
	A0-2	FH *	83.91%!	81.84%	-2.07%
	A3-9	T =	4.58%!	2.29%	-2.29%
	A3-9	H =	14.5%	22.14%!	7.634%
	A3-9	FT =	19.85%	29.01%!	9.16%
	A3-9	FH =	29.01%	42.75%!	13.74%
English	A0-2	T **	**75.47%!**	72.46%	-3.014%
	A0-2	H **	83.28%	**86.79%!**	3.511%
	A0-2	FT **	84.58%	**86.37%!**	1.782%
	A0-2	FH **	87.44%	**88.3%!**	0.8645%
	A3-9	T **	**28.55%!**	22.15%	-6.396%
	A3-9	H **	50.08%	**70.67%!**	20.59%
	A3-9	FT **	48.99%	**62.56%!**	13.57%
	A3-9	FH **	65.21%	**78.47%!**	13.26%
o-English	A0-2	T **	**74.34%!**	69.1%	-5.242%
	A0-2	H =	75.96%	77.52%!	1.563%
	A0-2	FT =	80.04%	80.39%!	0.3528%
	A0-2	FH =	81.4%	82.11%!	0.7056%
	A3-9	T =	5.714%	17.14%!	11.43%
	A3-9	H =	8.571%	17.14%!	8.571%
	A3-9	FT =	17.14%	22.86%!	5.714%
	A3-9	FH =	25.71%	40%!	14.29%
total	A0-2	T	*4, **4	*0, **0	-3.696%
	A0-2	H	*0, **0	*3, **2	1.907%
	A0-2	FT	*0, **0	*1, **1	0.1378%
	A0-2	FH	*2, **0	*1, **1	-0.1332%
	A3-9	T	*1, **1	*0, **0	-4.462%
	A3-9	H	*0, **0	*2, **2	15.03%
	A3-9	FT	*0, **0	*2, **2	14.12%
	A3-9	FH	*0, **0	*1, **1	13.94%
	A0-2	all	*6, **4	*5 ,**4	-0.4459%
	A3-9	all	*1, **1	*5 ,**5	9.658%
	A0-2	H,FT,FH	*2, **0	*5 ,**4	0.6372%
	A3-9	H,FT,FH	*0, **0	*5 ,**5	14.364%

It also seems that single-argument sub-tree settings sightly benefits to the majority classes. This is probably due to a better sub-tree selection.

The conclusion is that using sub-trees of a single argument makes the equivalence classes to have more samples but of better quality.

8.3 All-at-once vs one-at-a-time

Another possible variant of the system would be using multiple arguments per sub-tree, and forcing the k-NN module to use the same panel for predicting all arguments of the same sub-tree.

The panel size will correspond to the equivalence class which will satisfy the individual requirement (no ties and having valid votes) for producing a prediction for each semantic relation to be predicted. The resulting panel size can be larger than what needs it for each individual semantic relation. For instance, assuming a query sub-tree with two semantic relations to be predicted A and B. A, can be predicted with a panel size of the first equivalence class, but not with the first two equivalence classes, and B, can be not be predicted with the first equivalence class. In such case, A and B will be predicted using the first three or more equivalence classes in All-at-once settings, but using first and second equivalence class for one-at-a-time settings.

Table 8.3 shows the results, it suggests that using the same panel for all arguments have a positive impact, but the differences are so small that they can be dismissed: less than 0.02% in average.

In all 40 comparison there is no experiment showing more than 1% difference in accuracy, and in all experiments on German data the McNemar test fail to identify 25 different predictions, which invalidates the test.

8.4 Impact of using equivalence classes in k-NN

This section attempts to measure the benefit of using equivalence classes in k-NN classification. The system is compared with a variant in which each equivalence class contains a single sub-tree. This was created almost randomly by sorting the neighbour samples by their distance and re-defining its distance

Table 8.3: One vs all arguments at a time

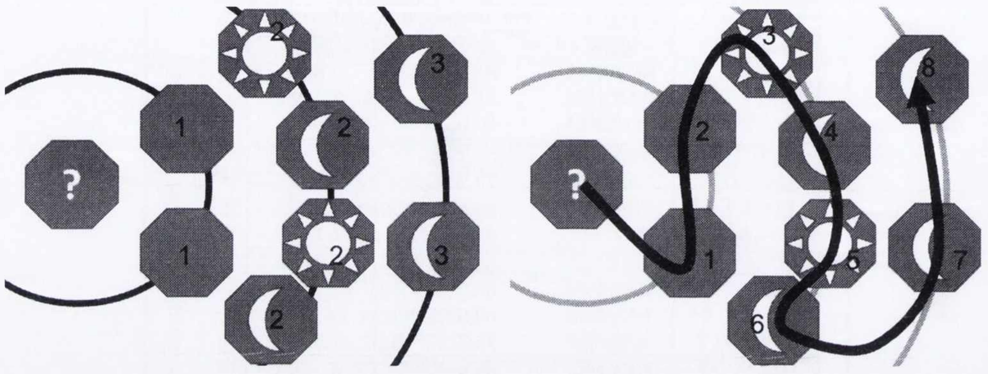
	accuracy	argument by argument: different panel for each argument	all arguments at the same time: same panel for all arguments	difference
Chinese	T *	83%	83.15%!	0.1443%
	H *	83.93%	84.09%!	0.1588%
	FT =	84.78%	84.82%!	0.03969%
	FH **	81.79%	**82.17%!**	0.3861%
German	T =	68.25%!	68.16%	-0.09311%
	H =	80.82%!	80.73%	-0.09311%
	FT =	86.03%!	85.57%	-0.4655%
	FH =	90.5%!	90.04%	-0.4655%
o-German	T =	69.1%	69.26%!	0.1675%
	H =	72.53%!	72.19%	-0.335%
	FT =	74.62%	74.62%!	0%
	FH =	77.89%!	77.72%	-0.1675%
English	T =	70.12%!	70.12%	-0.004294%
	H =	77.57%!	77.51%	-0.06012%
	FT *	77.98%!	77.7%	-0.2834%
	FH =	79.86%	80%!	0.146%
o-English	T =	65.77%!	65.59%	-0.1748%
	H =	66.68%!	66.47%	-0.2098%
	FT =	69.83%	69.83%!	0%
	FH =	69.69%!	69.23%	-0.4545%
Catalan	T =	65.23%!	65.13%	-0.09755%
	H =	74.76%	74.94%!	0.1774%
	FT **	**76.77%!**	76.37%	-0.4079%
	FH =	78.51%	78.52%!	0.008868%
Spanish	T *	64.89%!	64.61%	-0.2791%
	H *	73.95%!	73.67%	-0.2791%
	FT **	**75.04%!**	74.43%	-0.6089%
	FH =	77.61%!	77.47%	-0.1353%
Czech	T =	65.84%	65.93%!	0.08923%
	H =	73.28%	73.35%!	0.07138%
	FT =	69.99%!	69.96%	-0.03314%
	FH =	72.74%	72.8%!	0.06374%
o-Czech	T *	64.73%	65.06%!	0.3241%
	H =	72.56%	72.74%!	0.1729%
	FT *	69.24%	69.46%!	0.2233%
	FH *	72.23%	72.51%!	0.2809%
Japanese	T *	56.69%	57.49%!	0.8087%
	H *	71.92%	72.39%!	0.4702%
	FT **	60.43%	**61.39%!**	0.9592%
	FH *	70.53%	71.04%!	0.5078%
total	T	*1, **0	*3, **0	0.08851%
	H	*1, **0	*2, **0	0.007349%
	FT	*3, **2	*2, **1	-0.05768%
	FH	*0, **0	*3, **1	0.01705%
	all	*5, **2	*10 ,**2	0.01381%

Table 8.4: 1-NN vs first neighbour on the list

accuracy		1-NN	one neighbour	difference
Chinese	T **	**83%!**	77.41%	-5.597%
	H **	**83.93%!**	81.27%	-2.656%
	FT **	**84.78%!**	81.29%	-3.489%
	FH **	**81.79%!**	80.08%	-1.71%
German	T **	**68.25%!**	55.77%	-12.48%
	H *	80.82%!	79.24%	-1.583%
	FT **	**86.03%!**	83.43%	-2.607%
	FH =	90.5%!	89.39%	-1.117%
o-German	T **	**69.1%!**	61.47%	-7.621%
	H **	**72.53%!**	68.26%	-4.271%
	FT **	**74.62%!**	69.93%	-4.69%
	FH *	77.89%!	76.21%	-1.675%
English	T **	**70.12%!**	63.83%	-6.295%
	H **	**77.57%!**	75.16%	-2.413%
	FT **	**77.98%!**	74.73%	-3.251%
	FH **	**79.86%!**	78.36%	-1.499%
o-English	T **	**65.77%!**	59.62%	-6.154%
	H **	**66.68%!**	64.27%	-2.413%
	FT **	**69.83%!**	65.49%	-4.336%
	FH **	**69.69%!**	67.8%	-1.888%
Catalan	T **	**65.23%!**	58.75%	-6.474%
	H **	**74.76%!**	69.76%	-5.002%
	FT **	**76.77%!**	72.98%	-3.796%
	FH **	**78.51%!**	76.23%	-2.279%
Spanish	T **	**64.89%!**	57.61%	-7.281%
	H **	**73.95%!**	69.07%	-4.879%
	FT **	**75.04%!**	70.72%	-4.313%
	FH **	**77.61%!**	75.75%	-1.852%
Czech	T **	**65.84%!**	58.25%	-7.597%
	H **	**73.28%!**	70.02%	-3.258%
	FT **	**69.99%!**	64.94%	-5.05%
	FH **	**72.74%!**	70.21%	-2.529%
o-Czech	T **	**64.73%!**	57.87%	-6.865%
	H **	**72.56%!**	69.09%	-3.472%
	FT **	**69.24%!**	64.4%	-4.84%
	FH **	**72.23%!**	70.29%	-1.93%
Japanese	T **	**56.69%!**	50.91%	-5.774%
	H **	**71.92%!**	69.89%	-2.031%
	FT **	**60.43%!**	55.48%	-4.946%
	FH **	**70.53%!**	68.18%	-2.351%
total	T	*10, **10	*0, **0	-7.213%
	H	*10, **9	*0, **0	-3.198%
	FT	*10, **10	*0, **0	-4.132%
	FH	*9, **8	*0, **0	-1.883%
	all	*39, **37	*0 ,**0	-4.107%

as the position on the sorted vector. Then it assigns different distances to samples that originally were placed at the same tree edit distance.

Without using equivalence classes, panels are extended by what is effectively a random walk throw the equivalence class.



(a) **with** equivalence classes, predicts *moon* (b) **without** equivalence c., predicts *sun*

Figure 8.5: Effect of equivalence classes in k-NN
Numbers indicate the order in which samples would be used by the system.

In the example at Figure 8.5, the settings which uses equivalence classes (see Figure 8.5a) would use the third equivalence class to make the prediction because the first has no valid votes and the second (with four items) produces a tie. In this example the prediction would be a "moon". For the system without k-NN module, (see Figure 8.5b), the curve line represents the positions inside a sorted vector. For this particular sorted vector, the first two item does not produce any prediction, and the third one predicts a "sun". At that point the system makes its prediction ignoring all other items.

Table 8.4 shows the results. As it was expected 1-NN performs better than just using a random neighbour in all experiments. In 37 of 40 experiments it performs better with a strictly significant difference.

As the atomic costs settings gets more complex the amount of neighbours of the first equivalence class reduces. This happens because the settings select samples in a smother way. Hence, it reduces the differences between using 1-NN and just the first neighbour on the list. However the difference for the most complex settings (Frame Hamming) are still remarkable (1.883% in average).

8.5 Conclusion

This chapter explores some small variations in the architecture of the system, like extracting one sub-tree for each argument instead of extracting one sub-tree for each predicate. This strategy makes smaller sub-trees because it includes less context, surprisingly the accuracy is higher than using more context (multiple arguments per sub-tree), this happens because this strategy benefits the minority classes with a very small impact in the majority classes.

Another variation is to predict all arguments at the same time, by using the same k-NN panel of alignments for all arguments, the differences are negligible (0.014% in average).

Another variation was to re-assign distances to the vector of neighbours to hide the concept of equivalence class. Therefore, in the new set of distances each of the new equivalence classes contains a single sample. As it was expected, the accuracy drops, showing that it is important to design a system aware of equivalent classes.

Chapter 9

Conclusions and future work

9.1 Findings

This section is a summary of the findings of this thesis described in the previous four chapters.

9.1.1 Regarding the four atomic measures:

- The differences in accuracy between majority label and shape indicate that the structural information contains very relevant information (Section 5.2 on page 77). This was confirmed when linear representations were contrasted with tree representations. For all languages and swap-settings, when the system works with the linear representation it gives substantially poorer results than the system works with the tree representation, with an exception for Japanese Hamming. (Section 5.4.1 on page 103) This it is observed with and without dependency relationships in the data set. (Section 5.4.2 on page 107).

- All four atomic measures clearly out-perform Majority Label and Shape, and on average Frame Hamming gives the best performance (Section 5.3.1 on page 79).

- The hypothesis about the hierarchy of atomic measures is more or less accomplished (see Section 4.7.1 on page 72): *Shape* < *Ternary* < *Hamming*, *Ternary < Frame Ternary* and *Hamming < Frame Hamming*. *Hamming < Frame Hamming* seems to be the hypothesis with more exceptions:

1. In Chinese, Czech and Japanese, Hamming settings score higher than Frame Hamming settings, this could be due to a side effect of adding cost for frame mismatch. It would lead to a larger demand of training samples per frame, otherwise bad samples from a same frame may be promoted. (Section 5.3.2 on page 93)
2. Another point of view can be that the set of frames for those languages is too large, perhaps the frame set is too detailed or each frame can have too many senses/meanings. (Section 5.3.2 on page 93)
3. The amount of samples in Czech (reduced version) and Japanese seems to be insufficient for Frame Hamming to work properly (Section 5.3.2 on page 93).
4. The under-performance of Frame Hamming on the Chinese data set is surprising because it performs worse than the other tree measures used. A possible explanation could be that the variety of the lexical features with the variability of the predicate frames was especially high. Having very high perplexity of Lemma labels and very high perplexity of predicate frames, makes the lexical features potential noise for the k-NN module, as they may too often be different. Therefore lexical features are not very useful to the system to discriminate useful samples from useless ones once there is already a cost for frame mismatch. (Section 5.3.2.3 on page 95)

- Frame Ternary and Frame Hamming show better correlation to each other than Ternary and Hamming. (Section 5.3.1.5 on page 88)

9.1.2 Regarding language differences:

- Chinese has the highest overall accuracy. (Section 5.3.1.6 on page 90)
- Japanese reports the worst results of all languages, especially for the two Ternary version settings. This is probably due to the fact that the

Japanese data gives 96.1% of syntactic dependencies the same dependency relationship (perplexity = 1.17), but also to a very high average of nodes per sub-tree, what makes it difficult to find an adequate mapping. (Section 5.3.1.7 on page 90).

- The outcomes for the Spanish and Catalan data sets are very similar to each other. This is probably due to the fact that for most content, one is a translation of the other and they were annotated in the same way with the same set of labels. (Section 5.3.1.8 on page 90).

- The best systems in CoNLL-2009 evaluation show relatively low results for Chinese and German in comparison to other languages. On the system proposed in this thesis, higher accuracy was achieved for those two languages, what indicates that the techniques are especially suitable for Chinese and German. (Section 5.3.6 on page 100)

9.1.3 Regarding Out-of-domain:

- The out-of-domain evaluation got $\sim$ 10% less accuracy than the in-domain in both hamming measures and slightly less in the ternary measures. This was expected because Ternary measures are completely independent of the lexical features, what makes them more independent of the domain. (Section 5.3.1.9 on page 91)

- It is remarkable that all measures in Czech language and Ternary measures in German failed to drop from in-domain to out-of-domain as much as they should (drops are around one per cent). In German it even seems to increase the accuracy ($\sim$2% in Ternary). One of the reasons for the German case could be that the accuracy of the in-domain data set was already very low relative to other measurements. The fact that Ternary settings does not use lexical features can explain why it is less affected by the change of domain. (Section 5.3.1.9 on page 91)

- The order of arguments of the out-of-domain data set is more similar to the training data set than the in-domain evaluation data set or the perplexity of frames of Czech is higher than that of any other language, suggesting that the training data set has a very broad topic scope and it may include the out-of-domain as a sub-topic (Section 5.3.1.9 on page 91). It can explain the small drop in accuracy on Czech from in-domain to out-of-domain.

9.1.4 Regarding the frame mismatch cost and its tuning

- The settings with a cost for frame mismatching report higher accuracy, with three exceptions on the Hamming versions, Chinese, Czech and Japanese (Section 5.3.3 on page 96), but banning swaps of predicate from different frames reduces accuracy. (Section 5.3.4 on page 98)
- The high performance of the Frame versions of German can be a side effect produced by a good performance of the core arguments, which are more predicate dependent and the fact of having a small set of arguments. (Section 5.3.7 on page 101)
- Increasing the frame cost on Frame Ternary has a singularity point at cost=2, because after this value swaps of predicate nodes of different frame are not allowed. The same happens to Frame Hamming at cost=1.5 because if the two nodes belong to different frame they are also very likely not to share the lexical labels. (Section 6.6 on page 153). German seems an exception because it is difficult to observe those singular points. The reason may be that this data set has a small variety of semantic arguments and sequence of arguments what makes it need fewer amount of samples of each frame to work well. Hence, German data may not depend on samples of other frames as much as the other languages (Section 6.6 on page 155).

9.1.5 Regarding the tuning of deletion/insertion costs

- Frame versions tend to start falling at higher deletion values than the simple versions as deletion cost becomes smaller (Section 6.2 on page 109)

- The ratio between the cost of deleting and inserting two nodes and their swap cost specifies singularity points where the accuracy changes in a remarkable way (Section 6.2 on page 109). For Ternary and Frame Ternary settings:

    1. deletion=1: This only applies to Frame Ternary: Banning swaps of predicates from different frames which also do not share POS and DepRel labels seems to have a slightly positive effect. (Section 6.2.2.1 on page 117)
    2. deletion=0.5: This only applies to Frame Ternary: Banning swaps of predicates from different frames even if they share POS and DepRel labels seems to produce an important reduction on the accuracy of the system. (Section 6.2.2.3 on page 119).
    3. deletion=0.5: Banning swaps where both labels are different reduce accuracy, but the impact is very small. (Section 6.2.1.1 on page 114).
    4. deletion=0.25: Forcing the system to only swap nodes where POS and DepRel labels are equal always reduce accuracy. (see Section 6.2.1.2 on page 115 for Ternary and Section 6.2.2.4 on page 119 of Frame Ternary)
    5. deletion=0: The ranking generated by tree distance is fundamental for a good performance labelling, in comparison to using a random set of alignments. (See Section 6.2.1.4 on page 116 for Ternary and Section 6.2.2.6 on page 120 for Frame Ternary)

- For Hamming and Frame Hamming settings a different set of singularity points was found:

    1. deletion=0.75: This only applies to Frame Hamming: Banning swaps of predicate nodes which belong to a different frame and do not share lexical the lexical labels has a negative impact on accuracy (Section 6.2.4.3 on page 124).
    2. deletion=0.5: Banning swaps of nodes that do not share any label always has a negative effect. (See Section 6.2.3.1 on page 121 for Hamming and Section 6.2.4.4 on page 126 for Frame Hamming).

3. deletion=0.375: Banning swaps of nodes which share one or none common labels also always has a negative effect. (See Section 6.2.3.2 on page 121 for Hamming and Section 6.2.4.5 on page 127 for Frame Hamming).
4. deletion=0.25: Banning swaps of nodes which do not share lexical features (Lemma and Form tend to swap at the same time) also has a very negative impact on accuracy (See Section 6.2.3.3 on page 122 for Hamming and Section 6.2.4.6 on page 127 for Frame Hamming)
5. deletion=0.125: Allowing only swaps of nodes which share all four labels, has a negative impact. (See Section 6.2.3.4 on page 122 for Hamming and Section 6.2.4.7 on page 128).
6. deletion=0: Like in Ternary: The ranking of generated by tree distance is fundamental for the correct labelling, in comparison to use a random set of alignments. (See section 6.2.3.5 on page 123 and 6.2.4.8 on page 128.)

9.1.6 Regarding the tuning weights of different labels

- Perfect balance between dependency relations and POS is not the optimal value for all languages but it is optimal for most of the languages or at least very close to the optimal value for others. Japanese is the most exceptional case because its data set lacks annotation of dependency relationships which makes optimal to use only POS, yet it still has a local maximum when the weight of both features is balanced. (Section 6.3 on page 133).
- In the case of having to choose between dependency relations or POS to build a SRL system, the use of dependency relationships leads to higher accuracy. (Section 6.3 on page 133).
- In almost all cases, perfect balance between lexical and syntactic features was optimal, which corresponds to the original definition of Hamming. The two great exceptions to this trend are Japanese and Frame Hamming Chinese. (Section 6.4 on page 140)

1. Chinese: As Frame Hamming performs worse than Frame Ternary, the optimal value removes all lexical features, making the settings of the system become Frame Ternary.
2. Japanese: As there are no dependency relationships for this language, the lexical features have more weight than the syntactic ones; nine times more weight in fact.

- In the case of having to choose between lexical or syntactic features to build a SRL system, the features which lead to higher accuracy depend on the language. (Section 6.4 on page 140)

9.1.7 Regarding sub-trees:

- There is an inverse correlation among all languages between accuracy and the amount of nodes in the sub-tree to be labelled. (Section 5.3.8 on page 102)
- Extracting sub-trees which contain one argument per predicate leads to higher accuracy than extracting one sub-tree with all its arguments (Section 8.2 on page 194).

9.1.8 Regarding the k-NN module:

- The value k=10 tends to lead to better results than k=1, however the differences between different k values are really small. (Section 6.5 on page 142)
- The k-NN module has an important impact on the results compared with just picking random neighbours placed at the same distance. (Section 8.4 on page 203)

9.1.9 Regarding theoretical issues:

- Any hierarchical clustering outcome achieved via Δ and any categorisation outcome using nearest-neighbours can be replicated via Θ, but not vice-versa. (Section 7.2.4 on page 174)

- For the group of settings tested, Tai-distance performs better than Tai-similarity. (Section 7.3.1 on page 179).
- For distance brute measures work better than normalised (Section 7.3.2.1 on page 187) but for similarity normalised versions work better than brute measures (Section 7.3.2.2 on page 191).

9.2 Demarcation of the findings

The fact that the data set from this thesis is based on the CoNLL-2009 evaluation does not mean that the work is only concerns the CoNLL-2009 evaluation.

Some conclusions are likely to be universal across all types of Semantic Role Labelling systems like: Banning the usage of samples from different frames decreases accuracy. From this conclusion it can expect that data annotated in PropBank style is easier to annotate for a SRL system than in FrameNet style.

Other conclusions are likely to be universal across most Natural language processing tasks e.g. The parsing information has an important impact on the performance of the system, experiments bases on parting sentences into trees outperform the ones based on string comparison. Another example is the conclusion that POS gives more useful information to the system than the labels of the dependency relations.

And finally, the conclusions on the comparison between tree-distance and tree-similarity should be of application not only for computational linguistics systems based on Tree edit distance, but to any other field of science where tree edit distance/similarity is used.

9.3 Future work

9.3.1 Stricter Tai-Mapping base algorithms

Tai (1979) mappings are not the only relevant mappings for the Tree-SRL, but they are the ones chosen to be used in this thesis.

One of the many alternative, Höchsmann et al. (2003) algorithm, produces alignments; which is a kind of Tai mapping with more restrictions than the original Tai ones.

Other algorithms and types of mappings are not explored in this thesis but left as open questions for future work.

This option is interesting due to its increment of restriction over Tai-mappings, but it was not explored in this thesis as a first option because the regular distance based on Tai-mappings are more popular in the literature.

9.3.2 Lexical comparisons

Hamming and Frame Hamming measures used the lexical information in a simpler way; by just looking to the form and lemma of the words and decide the cost depending on whether the labels are identical or not. It is common in the literature to find very sophisticated lexical comparison measures which take into account if the words are synonyms or their frequency.

There is risk that this module can increase the complexity of the system too much making the task of analysing results difficult, but it is expected that it will lead to better accuracy.

This open question was mentioned in Section 5.3.1.2 on page 84.

9.4 Open questions

As open questions I would like to leave for future research the following:

- How much would the system improve if it had a module for word similarity to make hamming smother atomic costs between words depending on their semantic similarity. Note this open question is already part of future work.

- The problem of argument identification and predicate sense disambiguation:
  It was part of the CoNLL-2009 evaluation and they are the last modules need it to complete a full SRL, system. It would be interesting to solve this problem though Tai-distance, Tai-similarity or even other tree algorithms not based on Tai-mappings.

- Chapter 7, defines "Alignment ordering" based on the order of alignments between two given fixed trees (See Definition 7.3 on page 162), it could be define an "N-Alignment ordering" based on the order of alignments between one fixed tree and one variable tree, and it could define a "P-Alignment ordering" based on the order of the alignments between pairs of trees. In that case the question would be if "N-Alignment ordering" is equivalent to "Neighbour ordering" (See Definition 7.4 on page 162) and if "P-Alignment ordering" is equivalent to "Pair ordering" (See Definition 7.5 on page 162).

- Also in Chapter 7 Section 7.2.3.3 on page 172 leave a partial proof (7.27) about if N-duality implies P-duality, completion of which is an open question, but the partial prove would be completed if it would be possible to find a sequence of equidistant trees for any given pair $< S, T >$, which is also given as a partial proof (7.28 on page 173) and leave it as open question for cases in which the atomic distance between two identical nodes is higher than one.

Bibliography

Agirre, E. and Martinez, D. (2001). Knowledge Sources for Word Sense Disambiguation. In Matoušek, V., Mautner, P., Mouček, R., and Taušer, K., editors, *Text, Speech and Dialogue SE - 1*, volume 2166 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin Heidelberg.

Babko-malaya, O. (2005). Propbank annotation guidelines. *Portfolio The Magazine Of The Fine Arts*, (9):1–38.

Baker, C., Ellsworth, M., and Erk, K. (2007). SemEval'07 Task 19: Frame Semantic Structure Extraction. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, number 6, pages 99–104. Association for Computational Linguistics.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Batagelj, V. and Bren, M. (1995). Comparing resemblance measures. *Journal of Classification*, 12(1):73–90.

Bernard, M., Boyer, L., Habrard, A., and Sebban, M. (2008). Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8):2611–2629.

Boas, H. C. (2002). Bilingual framenet dictionaries for machine translation. In *Proceedings of the Third International Conference con Language Resources and Evaluation (LREC)*, number 1982, pages 1364–1371, Las Palmas de Gran Canaria, Spain.

Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER Treebank. In Hinrichs, E. W. and Simov, K., editors, *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning CoNLLX 06*, number June in CoNLL-X '06, pages 149–164. Association for Computational Linguistics.

Burchardt, A., Erk, K., and Frank, A. (2006). The SALSA corpus: a German corpus resource for lexical semantics. In *In Proceedings of LREC 2006*, number 1, pages 969–974.

Carreras, X., Màrques, L., and Màrquez, L. (2004). Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2004*, pages 89–97. Boston, MA, USA.

Carreras, X. and Màrquez, L. L. (2005). Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan. Association for Computational Linguistics.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics ACL 05*, 1(6):173–180.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Che, W., Li, Z., Li, Y., Guo, Y., Qin, B., and Liu, T. (2009). Multilingual Dependency-based Syntactic and Semantic Parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 49–54, Boulder, Colorado. Association for Computational Linguistics.

Chinchor, N. and Robinson, P. (1998). MUC-7 Named Entity Task Definition. *In Proceedings of the 7th Message Understanding Conference MUC7*, pages 1–7.

Civit, M., Martí, M. A., and Bufí, N. (2006). Cat3LB and Cast3LB: From Constituents to Dependencies. In *Advances in Natural Language Processing, Lecture notes in Computer Science, 5th International Conference on NLP, FinTAL 2006 ,August 23-25, Proceedings*, volume 4139, pages 141–152. Springer-Verlag, Turku, Finland.

Čmejrek, M., Hajič, J., and Kuboň, V. (2004). Prague Czech-English dependency treebank: Syntactically annotated resources for machine translation. In *In Proceedings of EAMT 10th Annual Conference*, pages 1597–1605.

Collins, M. J. (1999). *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia.

Dai, Q., Chen, E., and Shi, L. (2009). An Iterative Approach for Joint Dependency Parsing and Semantic Role Labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 19–24, Boulder, Colorado. Association for Computational Linguistics.

Demaine, E. D., Mozes, S., Rossman, B., and Weimann, O. (2009). An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms*, 6(1):1–19.

Ellsworth, M. and Janin, A. (2007). Mutaphrase: paraphrasing with FrameNet. In *In Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, number 6, pages 143–150. Association for Computational Linguistics.

Emms, M. (2005a). Adapting Tree Distance to Answer Retrieval and Parser Evaluation. Number 1, pages 53–59. MLMTA.

Emms, M. (2005b). Tree distance in answer retrieval and parser evaluation. In *Proceedings of NLUCS*, volume 2005, pages 100–108. Association for Computational Linguistics.

Emms, M. (2006a). Clustering by tree distance for parse tree normalisation. *Proceedings of NLUCS 2006*, pages 91–100.

Emms, M. (2006b). Variants of tree similarity in a Question Answering task. In *Proceedings of the Workshop on Linguistic Distances*, pages 100–108. Association for Computational Linguistics.

Emms, M. (2008). Tree-distance and some other variants of evalb. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, D. T., editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 1373–1379, Marrakech, Morocco. European Language Resources Association (ELRA).

Emms, M. and Franco-Penya, H.-H. (2012). On order equivalences between distance and similarity measures on sequences and trees. In *ICPRAM (1)*, pages 15–24.

Fillmore, C. J., Wooters, C., and Baker, C. (2001). Building a large lexical databank which provides deep semantics. In *Proceedings of the Pacific Asian Conference on Language, Information and Computation*, pages 3–25. Hong Kong.

Francis, W. N. and Kučera, H. (1979). Manual of information to accompany A standard corpus of present-day edited American English, for use with digital computers.

Furstenau, H. and Lapata, M. (2011). Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171.

Gesmundo, A., Henderson, J., Merlo, P., and Titov, I. (2009). A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 37–42, Boulder, Colorado. Association for Computational Linguistics.

Gildea, D. and Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Giménez, J. and Màrquez, L. (2007). Linguistic features for automatic evaluation of heterogenous MT systems. In *Proceedings of the Second Workshop on Statistical Machine Translation*, number June, pages 256–264, Prague, Czech Republic. Association for Computational Linguistics.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences: computer science and computational biology.* Cambridge Univ. Press.

Hajic, J. (2005). Complex Corpus Annotation: The Prague Dependency Treebank. *Insight into the Slovak and Czech Corpus Linguistics*, pages 54–78.

Hajic, J. (2008). Web page, CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages http://ufal.mff.cuni.cz/conll2009-st/ date accessed: 30-May-2013.

Hajic, J., Ciaramita, M., Johansson, R., Kawahara, D., Meyers, A., Nivre, J., Surdeanu, M., Xue, N., Zhang, Y., Hajič, J., Martí, M. A., Màrquez, L., Padó, S., Štěpánek, J., Stranák, P., and Marquez, L. (2009). The CoNLL-2009 shared task: syntactic and semantic dependencies in multiple languages. In *CoNLL '09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, number 6, pages 1–18, Morristown, NJ, USA. Association for Computational Linguistics.

Herrbach, C., Denise, A., Dulucq, S., and Touzet, H. (2006). Alignment of RNA secondary structures using a full set of operations. Technical Report 145, LRI.

Höchsmann, M., Töller, T., Giegerich, R., and Kurtz, S. (2003). Local similarity in RNA secondary structures. *Proceedings / IEEE Computer Society Bioinformatics Conference. IEEE Computer Society Bioinformatics Conference*, 2:159–68.

Isert, C. (1999). The Editing Distance Between Trees, Institut für Informatik, Technische Universität München. Technical report.

Johansson, R. and Nugues, P. (2007). Extended Constituent-to-Dependency Conversion for English. In *In Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, pages 105–112.

Kawahara, D., Kurohashi, S., and Hasida, K. (2002). Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 2008–2013.

Kendall, M. G. (1945). The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251.

Khoo, C. S.-G. (1997). The Use of Relation Matching in Information Retrieval. *LIBRES Library and Information Science Research Electronic Journal*, 1058(2):ISSN, 6768.

Klein, P. N. (1998). Computing the Edit-Distance Between Unrooted Ordered Trees. In *In Proceedings of the 6th Ann. European Symp. on Algorithms (ESA), - ESA'98*, pages 91–102, Springer Berlin Heidelberg.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 189–192.

Kolar, M., Lässig, M., and Berg, J. (2008). From protein interactions to functional annotation: graph alignment in Herpes. *BMC systems biology*, 2(1):90–99.

Kouylekov, M. and Magnini, B. (2005). Tree edit distance for textual entailment. In *Recent Advances in Natural Language Processing IV: Selected Papers from RANLP*, pages 17–20, Southampton, UK. In Proceedings of PASCAL Workshop on Recognizing Textual Entailment.

Kuboyama, T. (2007). Matching and Learning in Trees.

Lesot, M.-J. and Rifqi, M. (2010). Order-based equivalence degrees for similarity and distance measures. In *Proceedings of the Computational intelligence for knowledge-based systems design, and 13th international conference on Information processing and management of uncertainty*, IPMU'10, pages 19–28, Berlin, Heidelberg. Springer-Verlag.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Li, B., Emms, M., Luz, S., and Vogel, C. (2009). Exploring multilingual semantic role labeling. In *CoNLL '09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 73–78, Morristown, NJ, USA. Association for Computational Linguistics.

Litkowski, K. C. (2004). SENSEVAL-3 T ASK Automatic Labeling of Semantic Roles 9208 Gue Road Damascus , MD 20872 The Senseval-3 Task Introduction. In *In Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, volume 1, pages 141–146.

Litkowski, K. C., Hargraves, O., Hall, B., and Road, H. (2007). SemEval-2007 Task 06 : Word-Sense Disambiguation of Prepositions Preparation of Datasets. In *In Proceedings of the 4th International Workshop on Semantic Evaluations*, number 6, pages 24–29. Association for Computational Linguistics.

Marchler-Bauer, A., Panchenko, A. R., Shoemaker, B. A., Thiessen, P. A., Geer, L. Y., and Bryant, S. H. (2002). CDD: a database of conserved domain alignments with links to domain three-dimensional structure. *Nucleic acids research*, 30(1):281–283.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1994). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Màrquez, L., Carreras, X., Litkowski, K. C., and Stevenson, S. (2008). Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

Màrquez, L., Villarejo, L., Martí, M. A., and Taulé, M. (2007). Semeval-2007 task 09: Multilevel semantic annotation of Catalan and Spanish. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, number June, pages 42–47. Association for Computational Linguistics.

McDonald, R., Lerman, K., and Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. *In Proceedings of the Tenth Conference on Computational Natural Language Learning - CoNLL-X '06*, pages 216–220.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Mehdad, Y. (2009). Automatic cost estimation for tree edit distance using particle swarm optimization. *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers on - ACL-IJCNLP '09*, (8):289–292.

Melli, G., Wang, W.-Y., Liu, Y., Kashani, M., Shi, Z., Gu, B., Sarkar, A., and Popowich, F. (2005). Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2005 Summarization Task. In *In Proceeding of Document Understanding Conference (DUC-2005)*, pages 103–110.

Meyers, A., Reeves, R., Macleod, C., and R (2004). The nombank project: An interim report. In *In HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, pages 24–31.

Moreda, P., Navarro, B., and Palomar, M. (2005). Using semantic roles in information retrieval systems. In *Natural Language Processing and Information Systems Lecture Notes in Computer Science*, volume 3513, 2005, pages 192–202. Springer Berlin Heidelberg.

Moschitti, A., Pighin, D., and Basili, R. (2008). Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Narayanan, S. and Harabagiu, S. (2004). Question Answering Based on Semantic Structures. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 693–701, Geneva, Switzerland. COLING.

Negri, M., Kouylekov, M., and Magnini, B. (2008). Detecting expected answers Relations through Textual Entailment. In *In Proceedings of Cicling 2008*, pages 532–543, Haifa, Israel.

Omhover, J.-F., Rifqi, M., and Detyniecki, M. (2006). Ranking Invariance Based on Similarity Measures in Document Retrieval. In *Adaptive Multimedia Retrieval*, pages 55–64.

Padó, S. and Lapata, M. (2005). Cross-lingual projection of role-semantic information. In *In Proceedings of the conference on Human Language Technology* *and Empirical Methods in Natural Language Processing*, pages 859–866. Association for Computational Linguistics.

Palmer, M., Gildea, D., and Xue, N. (2010). *Semantic Role Labeling*. Morgan and Claypool publishers, Toronto.

Palmer, M., Kingsbury, P., and Gildea, D. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Pawlik, M. A. N. (2011). RTED: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345.

Ponzetto, S. P. and Strube, M. (2006). Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* -, 33(6):192–199.

Pradhan, S. S., Loper, E., Dligach, D., and Palmer, M. (2007). SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. In *In Proceedings of the 4th International Workshop on Semantic Evaluations*, number 6, pages 87–92. Association for Computational Linguistics.

Punyakanok, V., Roth, D., and Yih, S. W.-t. (2004a). Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004*, pages 1–10, Ford.

Punyakanok, V., Roth, D., and Yih, W.-t. (2004b). Natural language inference via dependency tree mapping: An application to question answering. *Computational Linguistics*, 6(9):1–10.

Ristad, E. S. and Yianilos, P. N. (1998). Learning String Edit Distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532.

Ruppenhofer, J., Sporleder, C., Morante, R., Baker, C., and Palmer, M. (2010). SemEval-2010 task 10: linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, number June 2009, pages 45–50, Morristown, NJ, USA. Association for Computational Linguistics.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In Jones, D., editor, *In Proceedings of international conference on new methods in language processing*, volume 12 of *Studies in Computational Linguistics*, pages 44–49. University of Stuttgart.

Sellers, P. H. (1974). On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793.

Shasha, D. and Zhang, K. (1990). Fast algorithms for the unit cost editing distance between trees. *J. Algorithms*, 11(4):581–621.

Shehata, S., Karray, F., and Kamel, M. (2008). Enhancing Text Categorization Using Sentence Semantics. In *Advanced Data Mining and Applications*, pages 87–98. Springer Berlin Heidelberg.

Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, number 6, pages 12–21.

Smith, T. F. and Waterman, M. S. (1981). Comparison of biosequences. *Advances in Applied Mathematics*, 2(4):482–489.

Spoustová, D. J., Hajič, J., Raab, J., and Spousta, M. (2009). Semi-supervised training for the averaged perceptron POS tagger. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics on EACL 09*, (4):763–771.

Surdeanu, M., Harabagiu, A., Williams, J., and Aarseth, P. (2003). Using Predicate-Argument Structures for Information Extraction. In *In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 8–15.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, number 8, pages 159–177. Association for Computational Linguistics.

Tai, K. K.-C. (1979). The tree-to-tree correction problem. *Journal of the ACM (JACM)*, 26(3):422–433.

Tatu, M. and Moldovan, D. (2005). A semantic approach to recognizing textual entailment. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, (10):371–378.

Taul, M., Antonia Martin, M., and Recasens, M. (2000). AnCora : Multilevel Annotated Corpora for Catalan and Spanish. In *In Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, pages 239–244.

Toutanova, K., Haghighi, A., and Manning, C. D. (2005). Joint learning improves semantic role labeling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 589–596, Morristown, NJ, USA. Association for Computational Linguistics.

Wagner, R. A. and Fischer, M. J. (1974). The String-to-String Correction Problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

Weischedel, R. and Brunstein, A. (2005). BBN Pronoun Coreference and Entity Type Corpus. Technical report, Linguistic Data Consortium, Philadelphia.

Went-tau Yih, S., Toutanova, K., and Yih, S. W.-t. (2006). Automatic Semantic Role Labeling. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 309–310. Association for Computational Linguistics.

Wikipedia (2012). wikipedia entry on Needleman algorithm date of last access: 18-June-2012. In *http://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_ algorithm.*

Wu, D. and Fung, P. (2009). Semantic Roles for SMT : A Hybrid Two-Pass Model. In *Proceedings of Human Language Technologies The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics Companion Volume Short Papers*, number June in NAACL-

Short '09, pages 13–16, Boulder, Colorado. Association for Computational Linguistics.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In *Proceedings of EMNLP*, volume 4, pages 88–94.

Xue, N. and Palmer, M. (2008). Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Xue, N., Xia, F., Chiou, F.-D., and Palmer, M. (2005). The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Yates, F. (1934). Contingency Tables Involving Small Numbers and the χ 2 Test. *Supplement to the Journal of the Royal Statistical Society*, 1(2):217–235.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.

Zielinski, A., Simon, C., and Wittl, T. (2009). Morphisto: Service-oriented open source morphology for German. In *In State of the Art in Computational Morphology*, pages 64–75. Springer Berlin Heidelberg.

Appendices

Appendix A

Data Set description

A.1 CoNLL-2009 data set

The CoNLL 2009 evaluation was a multilingual evaluation for seven languages: Catalan, Chinese, Czech, English, German, Japanese and Spanish. There is no single institution that annotated all languages with a uniform annotation. Each data set is described independently in this appendix.

A.1.1 English

A.1.1.1 English data: Compilation

The corpus was generated through merging the following corpora:

- **Penn TreeBank 3:** Marcus et al. (1994) hand coded parses of the Wall Street Journal (used in the test, development and training data set) and a small subset of the Brown corpus (Francis and Kučera, 1979) (used only on the test data set).

- **BBN Pronoun Co-reference and Entity Type Corpus** Weischedel and Brunstein (2005) annotates Wall Street Journal.

  Named entity categories include: Person, Organisation, Location, GPE, Facility, Money, Percent, Time and Date, based on the definitions of these categories in MUC (Chinchor and Robinson, 1998).

From this corpus only Name Entities boundaries were used to derive dependencies between Name Entities tokens, for example to create a NAME dependency from 'Mary' to 'Smith' given the NE mentioned 'Mary Smith'.

- **PropBank I** (Palmer et al., 2005) is already explained in section 2.2.2.
- **NomBank** (Meyers et al., 2004) uses the same framework as PropBank to annotate arguments of nouns.

A.1.1.2 English data: syntactic annotation

Originally Penn TreeBank Marcus et al. (1994) was written in constituent tree form, but in CoNLL-2009 used a dependency tree structure. The conversion from constituent tree to dependency tree requires identifying and labelling head-dependent relationships.

Head-dependent pairs are created from constituents by selecting one word in each phrase as the head and setting the others as its dependants.

This process was done using Johansson and Nugues (2007) algorithm, which is optimised for the Penn TreeBank annotation.

A.1.1.3 English data: semantic annotation

The English data set is part of the PropBank for which semantic annotation was already described above.

The out-of-domain evaluation data come from the English side of the Prague Czech-English Dependency Treebank (Čmejrek et al., 2004). LEMMA, POS and FEAT features were not annotated in the original data set, so predicted values were used.

A.1.2 Spanish and Catalan

A.1.2.1 Spanish and Catalan data: Compilation

The Catalan and Spanish datasets (Taul et al., 2000) were generated from the AnCora corpora, and the annotation is identical for the two languages.

The Catalan data set is compiled from the EFE Catalan news agency (around 75 thousand words), ACN Catalan news agency (around 225 thousand words), and 'El Periodico' newspaper (around 200 thousand words).

The Spanish data set is compiled from the EFE Spanish news agency (around 225 thousand words), and the Spanish version of 'El Periodico' (around 200 thousand words) which corresponds to the same news in Catalan and Spanish, spanning from January to December 2000.

Part of the Catalan and Spanish data set is a translation and each other, which corresponds to the segment of 'El periodico'.

A.1.2.2 Spanish and Catalan data: Syntactic annotation

The transformation from constituent tree structures to dependency tree structures uses Civit et al. (2006) algorithm.

The predicted features PLEMMA, PPOS, and PFEAT had been predicted with the FreeLing Open source suite of Language Analysers in which PLEMA and PPOS accuracy is above 95%, and PHEAD and PDEPREL had been predicted by MaltParser, with an accuracy above 86% for both languages (Hajic et al., 2009).

A.1.2.3 Spanish and Catalan data: Semantic annotation

The linguistic annotation is identical for both languages, and as in English, the core arguments are enumerated arg1 to arg4 and defined per each predicate. argM (for adjuncts), argL (for complements of light verbs).

In addition the tags contain a thematic role. There are 20 thematic roles:

- AGT: Agent,
- AGI: Induced Agent,
- CAU: Cause,
- EXP: Experiencer,
- SCR: Source,
- PAT: Patient,

- TEM: Theme,
- ATR: Attribute,
- BEN: Beneficiary,
- EXT: Extension,
- INS: Instrument,
- LOC: Locative,
- TMP: Time,
- MNR: Manner,
- ORI: Origin,
- DES: Goal,
- FIN: Purpose,
- EIN: Initial State,
- EFI: Final State,
- ADV: Adverbial.

A.1.2.4 Spanish and Catalan data: Important considerations

The amount of long sentences is remarkable on this corpus. In the Catalan corpora 10.73% of the sentences are longer than 50 tokens, and 4.42% are longer than 60 (Hajic et al., 2009)

For Hajic et al. (2009) the more remarkable features of the Catalan and Spanish data sets (CoNLL-2009) are:

1. all dependency trees are projective;
2. no word can be the argument of more than one predicate in a sentence;
3. semantic dependencies completely match syntactic dependency structures (no new edges are introduced by the semantic structure);

4. only verbal predicates are annotated (with exceptional cases referring to words that can be adjectives and past participles);
5. the corpus is segmented so multi-words, named entities, temporal expressions, compounds, etc. are grouped together; and
6. segmentation also accounts for elliptical pronouns (there are marked as empty lexical tokens '˛ with a pronoun POS tag).

A.1.3 Chinese

A.1.3.1 Chinese data: Compilation

The Chinese Corpus merged the Chinese TreeBank 6.0 (Xue et al., 2005) and the Chinese Proposition Bank 2.0 (Xue and Palmer, 2008).

The TreeBank is composed from Xinhua newswire (mainland China), HongKong news, and Sinorama Magazine (Taiwan). The most recent expansion under DARPA GALE funding includes broadcast news, broadcast conversation, news groups and web-log data. The version of Chinese treebank used on this thesis is CTB 6.0, it includes newswire, magazine articles, and transcribed broadcast news 12.

A.1.3.2 Chinese data: Syntactic annotation

The conversion from constituent tree structures to dependency tree structures was done by the organisers of CoNLL2009, identifying the head of each constituent by analysing the structural information and detecting six broad categories of syntactic relationships (predication, modification, complementation, coordination, auxiliary, and flat).

A.1.3.3 Chinese data: Semantic annotation

As in the English data set, the Chinese Proposition Bank annotates the core arguments are numerate from zero to five and defined for each predicate. The version of Chinese Proposition Bank used on this thesis is CPB 2.0, excluding nominal predicates, thus its annotation was incomplete.

A.1.4 Czech

A.1.4.1 Czech data: Compilation

The Czech corpus is extracted from the Prague Dependency Treebank 2.0 (Hajic, 2005). which annotates the Prague Dependency Treebank using the following new papers:[1]

- Lidové noviny (daily newspapers), ISSN 1213-1385, 1991, 1994, 1995
- Mladá fronta Dnes (daily newspapers), 1992
- Českomoravský Profit (business weekly), 1994
- Vesmír (scientific journal), ISSN 1214-4029, Vesmír, s.r.o., 1992, 1993

The out-of-domain evaluation data comes from the Czech side of the Prague Czech-English Dependency Treebank (Čmejrek et al., 2004), just like the English out-of-domain evaluation data.

A.1.4.2 Czech data: Syntactic annotation

For the out-of-domain data: LEMMA, POS and FEAT features were not annotated in the original data set, so predicted values were used.

FORM was annotated using the "form" element of the morphological layer.

LEMMA was annotated also using the "lemma" element on the morphological layer, taking only the initial strings. Therefore, the is no difference between homonyms.

The POS column was annotated using the morphological "tag" element, taking the first character.

The remaining characters of "tag" plus the special feature "Sem" corresponding to a semantic feature of lemma were used to annotated FEAT.

HEAD was annotated using the PDT ord attribute.

The predicted features, PLEMMA, PPOS and PFEAT were generated by the (cross-trained) morphological tagger MORCE (Spoustová et al., 2009)

[1]http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch03.html#a-data-sources

A.1.4.3 Czech data: Semantic annotation

The PRED and APREDs columns were annotated from the tectogrammatical layer of PDT 2.0 and the valency lexicon PDT-Vallex. The details about the rules used can be found at (Hajic et al., 2009).

A.1.5 German

A.1.5.1 German data: Compilation

The German corpus is extracted from SALSA (Burchardt et al., 2006) (TIGER newspaper corpus (Brants et al., 2002)) which annotates the articles from the German newspaper *Frankfurter Rundschau* from all kind of domains.

The out-of-domain dataset was taken from (Padó and Lapata, 2005), which is a sample of EUROPARL corpus.

A.1.5.2 German data: Syntactic annotation

The conversion algorithm to dependency structures was borrowed from the organisers of CoNLL-X (Buchholz and Marsi, 2006) with minor modifications.

The predicted PLEMMA and PPOS was produce by Tree Tagger algorithm (Schmid, 1994), and PFEAT was predicted by Morphisto morphology (Zielinski et al., 2009).

A.1.5.3 German data: Semantic annotation

Salsa is annotated with semantic roles from a FrameNet paradigm. Hence, the CoNLL2009 organiser mapped the semantic roles to PropBank arguments at the level of each frame.

The original annotation of SALSA was considerably simplified, for example: multi-word expressions annotation, annotations involving multiple frames for the same predicate and inter-sentence roles were removed.

A.1.6 Japanesse

A.1.6.1 Japanese data: Compilation

The Japanese data set come from the Kyoto University Text Corpus (Kawahara et al., 2002) It contains approximately 40.000 sentences taken from *Mainichi* Newspapers. Nearly 5.000 sentences are annotated with syntactic and semantic dependencies, and are used the training, development and test data set

The other 35.000 only have syntactic annotation and it was used for the training corpus.

A.1.6.2 Japanese data: Syntactic annotation

The original data set was annotated with dependency relationships based on phrases (Japanese *bunsetsu*). Therefore it was necessary to convert the data to word-based dependency. The criteria for the conversion was the following: all words except the last word in a phrase depend on the next (right) word, and the last word in a phrase basically depends on the head word of the governing phrase.

PLEMA, PPOS and PFEAT were predicted using the morphological analyser JUMAN [2] and the dependency and case structure analyser KNP [3]. PHEAD and PDEPREL were predicted using the MSTParser [4]

A.1.6.3 Japanese data: Semantic annotation

There are semantic annotations for predicate verbs and predicate nouns.

There are 41 surface cases, “ga” indicates nominative, “wo” indicates accusative and “ni” indicates dative. Semantic frame discrimination is not annotated. Therefore, PRED contains the same annotation as LEMMA. The original data set contains co-reference annotations and inter-sentence semantic relations which were not used.

[2]http:// nlp.kuee.kyoto-u.ac.jp/ nl-resource/juman-e.html
[3]http:// nlp.kuee.kyoto-u.ac.jp/ nl-resource/knp-e.html
[4]http://sourceforge.net/ projects/ mstparser

A.2 Semantic Labels description

The symbol "|" is a separator that indicates an extra label for a semantic relationship, in this case the system have to predict both labels in order to count the prediction as correct.

A.2.1 Catalan and Spanish

Amount labels: 45	CA-T	CA-E	CA-D	ES-T	ES-E	ES-D
perplexity	11.12	11.15	10.93	11.22	11.05	11.08
total amount	84367	11275	11529	99054	11824	11600
arg1-pat	2.2022e+01%	2.1898e+01%	2.2682e+01%	2.0467e+01%	2.0171e+01%	2.0534e+01%
arg0-agt	1.8308e+01%	1.7845e+01%	1.8267e+01%	1.8880e+01%	1.9181e+01%	1.9216e+01%
arg1-tem	1.4502e+01%	1.4271e+01%	1.4268e+01%	1.4723e+01%	1.4547e+01%	1.4293e+01%
argM-tmp	8.3872e+00%	8.5144e+00%	8.1534e+00%	8.1400e+00%	9.0240e+00%	8.7500e+00%
arg2-atr	8.1015e+00%	7.6896e+00%	7.6329e+00%	8.2046e+00%	7.9584e+00%	7.7500e+00%
argM-adv	5.9917e+00%	6.0931e+00%	6.5400e+00%	7.6978e+00%	7.9753e+00%	8.0776e+00%
argM-loc	5.5910e+00%	6.1197e+00%	5.3951e+00%	4.9953e+00%	5.0152e+00%	4.8276e+00%
arg2-null	3.2572e+00%	3.6009e+00%	3.4175e+00%	2.4825e+00%	2.1820e+00%	2.5086e+00%
arg2-ben	1.7874e+00%	1.9690e+00%	2.0644e+00%	2.2462e+00%	2.0974e+00%	2.3017e+00%
arg1-null	1.7175e+00%	1.8803e+00%	1.7087e+00%	1.7657e+00%	1.5562e+00%	1.7586e+00%
argM-cau	1.4958e+00%	1.6940e+00%	1.4052e+00%	1.5799e+00%	1.4716e+00%	1.5776e+00%
argM-fin	1.5516e+00%	1.5344e+00%	1.7174e+00%	1.4164e+00%	1.3447e+00%	1.4224e+00%
arg2-loc	1.3619e+00%	1.3836e+00%	1.2577e+00%	1.6042e+00%	1.3955e+00%	1.5948e+00%
argM-mnr	1.5563e+00%	1.5344e+00%	1.4138e+00%	1.3710e+00%	1.6407e+00%	1.2759e+00%
arg0-cau	1.0715e+00%	9.3126e-01%	1.1363e+00%	1.2205e+00%	1.4124e+00%	1.1207e+00%
argM-atr	8.8779e-01%	8.1596e-01%	8.7605e-01%	6.5217e-01%	7.3579e-01%	7.3276e-01%
arg4-des	5.0138e-01%	4.7007e-01%	3.9032e-01%	6.2693e-01%	5.2436e-01%	5.1724e-01%
argL-null	4.7886e-01%	4.3459e-01%	4.4236e-01%	4.7247e-01%	3.8904e-01%	3.7069e-01%
arg2-ext	3.0344e-01%	2.4834e-01%	3.2960e-01%	2.8772e-01%	2.7909e-01%	2.5862e-01%
arg3-ori	1.7661e-01%	3.1042e-01%	1.0409e-01%	2.3422e-01%	1.7760e-01%	1.2931e-01%
arg2-efi	1.5172e-01%	2.1286e-01%	1.0409e-01%	1.8273e-01%	2.0298e-01%	2.1552e-01%
arg3-ben	1.2090e-01%	1.2417e-01%	2.0817e-01%	1.4739e-01%	1.2686e-01%	1.2931e-01%
arg4-efi	1.5646e-01%	1.5965e-01%	1.5613e-01%	9.4898e-02%	1.6915e-01%	1.3793e-01%
argM-ext	1.0075e-01%	8.8692e-03%	2.6021e-02%	1.2317e-01%	1.6069e-01%	9.4828e-02%
arg2-exp	1.0194e-01%	2.6608e-02%	7.8064e-02%	8.6821e-02%	5.0744e-02%	1.3793e-01%
arg3-ein	7.3488e-02%	5.3215e-02%	5.2043e-02%	5.2497e-02%	7.6116e-02%	3.4483e-02%
arg3-fin	-	-	-	7.5716e-02%	3.3829e-02%	5.1724e-02%
arg1-ext	4.0300e-02%	4.4346e-02%	2.6021e-02%	2.9277e-02%	4.2287e-02%	2.5862e-02%
arg2-fin	6.7562e-02%	6.2084e-02%	7.8064e-02%	-	-	-
arg1-loc	1.7779e-02%	8.8692e-03%	-	3.4325e-02%	8.4574e-03%	4.3103e-02%
arg2-ins	1.4224e-02%	-	1.7348e-02%	1.9181e-02%	1.6915e-02%	8.6207e-03%
arg3-exp	1.8965e-02%	-	8.6738e-03%	1.3124e-02%	-	5.1724e-02%
arg0-exp	1.3038e-02%	-	-	1.9181e-02%	8.4574e-03%	8.6207e-03%
argM-ins	5.9265e-03%	8.8692e-03%	-	2.1201e-02%	8.4574e-03%	8.6207e-03%
arg0-src	7.1118e-03%	-	8.6738e-03%	1.6153e-02%	1.6915e-02%	8.6207e-03%
arg3-atr	1.7779e-02%	3.5477e-02%	1.7348e-02%	1.0096e-03%	-	8.6207e-03%
arg3-loc	1.7779e-02%	1.7738e-02%	8.6738e-03%	3.0287e-03%	-	-
arg3-null	1.6594e-02%	-	-	1.0096e-03%	-	-
arg3-ins	7.1118e-03%	-	-	5.0478e-03%	-	-
arg0-null	-	-	-	2.0191e-03%	-	1.7241e-02%
arg2-tem	-	-	-	2.0191e-03%	-	-
arg1-tmp	-	-	8.6738e-03%	-	-	-
arg0-pat	-	-	-	1.0096e-03%	-	-
argM-LOC	-	-	-	1.0096e-03%	-	-
arg3-des	-	-	-	1.0096e-03%	-	-

Table A.1: Catalan and Spanish semantic labels

A.2.2 English

Amount labels: 53	T	E	D	ODD
perplexity	7.121	7.189	7.118	7.782
total amount	393699	23286	13865	2859
A1	3.7223e+01%	3.7057e+01%	3.7663e+01%	3.5327e+01%
A0	2.5245e+01%	2.5148e+01%	2.4854e+01%	2.6128e+01%
A2	1.1872e+01%	1.1312e+01%	1.1504e+01%	7.9398e+00%
AM-TMP	5.9302e+00%	6.7251e+00%	6.3325e+00%	4.2672e+00%
AM-MNR	3.0066e+00%	2.9288e+00%	3.0941e+00%	4.6170e+00%
AM-LOC	2.6383e+00%	2.5509e+00%	2.4594e+00%	3.0430e+00%
A3	2.3955e+00%	2.2245e+00%	2.1781e+00%	6.2959e-01%
AM-MOD	2.2944e+00%	2.3491e+00%	2.2719e+00%	3.1829e+00%
AM-ADV	2.1694e+00%	2.2417e+00%	2.0700e+00%	5.1067e+00%
AM-DIS	1.2256e+00%	1.3742e+00%	1.4569e+00%	7.6950e-01%
R-A0	1.0259e+00%	9.6195e-01%	1.0386e+00%	8.7443e-01%
AM-NEG	9.2787e-01%	1.1251e+00%	8.8713e-01%	2.0637e+00%
A4	7.7826e-01%	5.0245e-01%	6.0584e-01%	5.5964e-01%
C-A1	6.9952e-01%	8.3741e-01%	1.0025e+00%	1.0493e+00%
R-A1	5.8598e-01%	6.6564e-01%	5.9142e-01%	7.3452e-01%
AM-PNC	5.6718e-01%	4.9386e-01%	5.7699e-01%	5.9461e-01%
AM-EXT	3.3249e-01%	2.8343e-01%	3.3898e-01%	2.0986e-01%
AM-CAU	3.0988e-01%	3.0920e-01%	3.3177e-01%	2.7982e-01%
AM-DIR	2.9109e-01%	3.6503e-01%	2.4522e-01%	1.8538e+00%
R-AM-TMP	1.7882e-01%	2.2331e-01%	2.2358e-01%	3.4977e-01%
R-A2	7.2644e-02%	6.8711e-02%	3.6062e-02%	-
R-AM-LOC	5.3848e-02%	9.0183e-02%	6.4912e-02%	1.3991e-01%
R-AM-MNR	3.5814e-02%	2.5767e-02%	4.3274e-02%	6.9955e-02%
A5	2.0828e-02%	2.5767e-02%	2.1637e-02%	-
AM-PRD	1.7526e-02%	2.1472e-02%	2.1637e-02%	3.4977e-02%
C-A0	1.6256e-02%	2.1472e-02%	-	3.4977e-02%
C-A2	1.5494e-02%	-	2.1637e-02%	3.4977e-02%
R-AM-CAU	1.0414e-02%	1.7178e-02%	2.1637e-02%	6.9955e-02%
C-A3	8.8900e-03%	8.5889e-03%	-	-
R-A3	7.1120e-03%	4.2944e-03%	-	-
C-AM-MNR	5.0800e-03%	4.2944e-03%	1.4425e-02%	3.4977e-02%
C-AM-ADV	5.0800e-03%	-	-	-
AM-REC	3.5560e-03%	8.5889e-03%	-	-
AA	3.3020e-03%	-	7.2124e-03%	-
R-AM-PNC	3.0480e-03%	-	-	-
C-AM-TMP	2.5400e-03%	4.2944e-03%	-	-
C-AM-EXT	2.7940e-03%	-	-	-
C-A4	2.7940e-03%	-	-	-
C-AM-LOC	2.2860e-03%	-	-	-
AM	2.2860e-03%	-	-	-
R-A4	1.7780e-03%	4.2944e-03%	-	-
R-AM-ADV	1.2700e-03%	8.5889e-03%	-	-
C-AM-DIR	1.2700e-03%	4.2944e-03%	7.2124e-03%	-
C-AM-CAU	1.5240e-03%	-	7.2124e-03%	-
R-AM-EXT	1.0160e-03%	4.2944e-03%	7.2124e-03%	-
C-AM-PNC	1.2700e-03%	-	-	-
C-AM-DIS	7.6200e-04%	-	-	-
AM-PRT	5.0800e-04%	-	-	-
R-AA	5.0800e-04%	-	-	-
AM-TM	5.0800e-04%	-	-	-
C-AM-NEG	2.5400e-04%	-	-	-
C-R-AM-TMP	2.5400e-04%	-	-	-
R-AM-DIR	2.5400e-04%	-	-	-

Table A.2: English semantic labels

A.2.3 Chinese

Amount labels: 37	T	E	D
perplexity	6.435	6.529	6.652
total amount	231869	27712	18554
A1	3.0426e+01%	3.0860e+01%	3.1249e+01%
A0	2.6556e+01%	2.5740e+01%	2.5693e+01%
ADV	1.9844e+01%	1.9295e+01%	1.8400e+01%
TMP	6.6141e+00%	7.2243e+00%	7.2006e+00%
DIS	4.4387e+00%	4.1390e+00%	3.9884e+00%
A2	3.5473e+00%	3.7276e+00%	3.7512e+00%
LOC	2.7964e+00%	3.0023e+00%	2.9589e+00%
MNR	1.4888e+00%	1.7934e+00%	1.9565e+00%
C-A0	9.1172e-01%	8.6244e-01%	9.8631e-01%
PRP	9.1819e-01%	7.7223e-01%	1.0025e+00%
C-A1	6.7797e-01%	8.6244e-01%	9.8631e-01%
DIR	3.8513e-01%	3.5003e-01%	2.8026e-01%
CND	3.6529e-01%	2.8147e-01%	3.8267e-01%
EXT	2.4712e-01%	2.2734e-01%	3.2338e-01%
A3	2.4410e-01%	2.8508e-01%	2.6409e-01%
TPC	2.4842e-01%	2.3456e-01%	1.8325e-01%
BNF	1.7553e-01%	2.0930e-01%	1.9403e-01%
A4	1.8976e-02%	3.2477e-02%	6.4676e-02%
FRQ	2.3289e-02%	2.1651e-02%	2.6948e-02%
C-A2	2.0701e-02%	2.5260e-02%	2.1559e-02%
PRD	1.6820e-02%	1.0826e-02%	5.9286e-02%
DGR	1.0782e-02%	1.4434e-02%	1.0779e-02%
ASP	6.9004e-03%	1.0826e-02%	-
QTY	3.0189e-03%	3.6085e-03%	1.0779e-02%
CRD	2.5877e-03%	1.0826e-02%	-
C-C-A0	3.4502e-03%	-	-
ARGM	1.7251e-03%	3.6085e-03%	-
PN	2.1564e-03%	-	-
VOC	1.2938e-03%	-	-
C-ARGM-TMP	1.2938e-03%	-	-
C-A3	8.6256e-04%	-	-
PSR	4.3128e-04%	-	-
A5	4.3128e-04%	-	-
C-ARGM-DIS	4.3128e-04%	-	-
T	4.3128e-04%	-	-
PSE	4.3128e-04%	-	-
C-ADV	-	-	5.3897e-03%

Table A.3: Chinese semantic labels

A.2.4 Czech

Amount labels: 64	T	E	D	ODD
perplexity	10.93	11.05	11.07	11.33
total amount	365255	39223	49071	13882
RSTR	3.0128e+01%	2.9977e+01%	2.9684e+01%	2.9081e+01%
PAT	1.8363e+01%	1.8522e+01%	1.8856e+01%	1.7526e+01%
ACT	1.6723e+01%	1.6442e+01%	1.6388e+01%	1.8059e+01%
APP	6.3832e+00%	6.1902e+00%	6.2073e+00%	4.1565e+00%
LOC	4.4287e+00%	4.4795e+00%	4.4425e+00%	3.3136e+00%
TWHEN	3.8614e+00%	3.9263e+00%	3.9025e+00%	3.5370e+00%
MANN	1.9841e+00%	2.1289e+00%	1.9849e+00%	1.7361e+00%
EXT	1.6736e+00%	1.6113e+00%	1.7994e+00%	3.1408e+00%
EFF	1.4910e+00%	1.4201e+00%	1.6425e+00%	3.5225e+00%
ADDR	1.3333e+00%	1.4609e+00%	1.3776e+00%	1.0013e+00%
DIR3	1.2326e+00%	1.2722e+00%	1.2431e+00%	8.3561e-01%
MAT	1.1272e+00%	1.1651e+00%	1.0291e+00%	2.0314e+00%
ID	1.0226e+00%	9.2548e-01%	9.2723e-01%	2.0890e+00%
DIR1	9.8041e-01%	9.1528e-01%	9.6595e-01%	8.9324e-01%
BEN	8.6323e-01%	9.2038e-01%	8.6609e-01%	5.8349e-01%
ACMP	7.5399e-01%	8.2605e-01%	7.3771e-01%	5.0425e-01%
REG	6.4640e-01%	7.2661e-01%	7.1325e-01%	8.3561e-01%
MEANS	5.8425e-01%	6.3993e-01%	5.9302e-01%	3.8179e-01%
CPHR	5.8480e-01%	5.7109e-01%	6.0525e-01%	4.0340e-01%
CRIT	5.3962e-01%	5.1500e-01%	5.7468e-01%	2.6653e-01%
CAUS	5.1854e-01%	5.5835e-01%	5.7875e-01%	5.5468e-01%
COND	5.2128e-01%	5.2520e-01%	5.7264e-01%	4.1060e-01%
AIM	5.1115e-01%	4.6911e-01%	4.9520e-01%	4.6103e-01%
THL	4.3449e-01%	4.1302e-01%	4.6260e-01%	7.7078e-01%
COMPL	3.1430e-01%	3.4928e-01%	3.5051e-01%	3.2416e-01%
THO	2.7488e-01%	2.6260e-01%	3.3829e-01%	2.4492e-01%
DPHR	2.4969e-01%	2.6260e-01%	2.7919e-01%	2.8814e-01%
TTILL	2.4394e-01%	2.4730e-01%	2.0990e-01%	2.0890e-01%
ORIG	2.1820e-01%	2.2181e-01%	1.8748e-01%	4.5383e-01%
COMPL2	2.2122e-01%	2.1161e-01%	2.3028e-01%	2.0170e-01%
DIFF	1.8946e-01%	2.0906e-01%	1.5895e-01%	4.6103e-01%
TSIN	1.8672e-01%	1.8866e-01%	1.9971e-01%	2.0170e-01%
CPR	1.8152e-01%	1.6317e-01%	1.7933e-01%	4.2501e-01%
AUTH	1.6728e-01%	1.8102e-01%	1.2431e-01%	7.2036e-02%
CNCS	1.5359e-01%	1.6062e-01%	1.4673e-01%	1.6568e-01%
RESTR	1.4839e-01%	1.1473e-01%	1.4876e-01%	5.0425e-02%
TPAR	1.3196e-01%	1.4277e-01%	1.1616e-01%	2.8814e-01%
DIR2	8.9253e-02%	1.0963e-01%	8.9666e-02%	2.1611e-02%
RESL	8.0218e-02%	8.6684e-02%	9.9855e-02%	1.2246e-01%
TFHL	7.6111e-02%	7.3936e-02%	8.5590e-02%	4.3221e-02%
SUBS	5.9684e-02%	5.0990e-02%	7.7439e-02%	5.7629e-02%
ACT|ADDR	5.3661e-02%	5.3540e-02%	4.2795e-02%	4.3221e-02%
ACT|PAT	5.0650e-02%	5.0990e-02%	5.2984e-02%	3.6018e-02%
CONTRD	4.1067e-02%	3.5693e-02%	2.4454e-02%	5.7629e-02%
INTT	4.0246e-02%	3.8243e-02%	3.8719e-02%	1.4407e-02%
ACT|COMPL	3.6960e-02%	5.0990e-02%	5.5022e-02%	-
TOWH	3.6687e-02%	5.3540e-02%	4.0757e-02%	3.6018e-02%
TFRWH	3.6139e-02%	3.5693e-02%	3.2606e-02%	6.4832e-02%
ADDR|PAT	1.1773e-02%	5.0990e-03%	1.0189e-02%	-
EFF|PAT	5.7494e-03%	1.5297e-02%	1.4265e-02%	-
HER	6.0232e-03%	1.2748e-02%	6.1136e-03%	-
ORIG|PAT	1.6427e-03%	2.5495e-03%	6.1136e-03%	-
COMPL|PAT	1.3689e-03%	-	2.0379e-03%	-
ACT|EFF	2.7378e-04%	2.5495e-03%	2.0379e-03%	-
ACMP|ACT	5.4756e-04%	-	-	-
FPHR	-	-	-	1.4407e-02%
ADDR|MANN	2.7378e-04%	-	-	-
MEANS|PAT	2.7378e-04%	-	-	-
ACT|ORIG	2.7378e-04%	-	-	-
ACT|AIM	2.7378e-04%	-	-	-
BEN|PAT	2.7378e-04%	-	-	-
ACT|DIR3	-	2.5495e-03%	-	-
ACT|REG	-	2.5495e-03%	-	-
ACT|TWHEN	-	-	-	7.2036e-03%

Table A.4: Czech semantic labels

A.2.5 German

Amount labels: 10	T	E	D	ODD
perplexity	3.579	3.742	3.736	3.771
total amount	34276	1073	1169	1193
A0	4.0393e+01%	3.8956e+01%	3.7810e+01%	3.7049e+01%
A1	3.9462e+01%	3.8211e+01%	3.8580e+01%	4.2498e+01%
A2	1.1836e+01%	1.2861e+01%	1.3430e+01%	9.4719e+00%
A3	5.8262e+00%	6.9897e+00%	7.6133e+00%	6.6220e+00%
A4	1.2808e+00%	1.9571e+00%	1.8820e+00%	1.8441e+00%
A5	6.6227e-01%	3.7279e-01%	8.5543e-02%	9.2205e-01%
A7	2.1006e-01%	2.7959e-01%	3.4217e-01%	1.0897e+00%
A6	1.8088e-01%	-	1.7109e-01%	-
A8	1.2837e-01%	3.7279e-01%	8.5543e-02%	5.0293e-01%
A9	2.0422e-02%	-	-	-

Table A.5: German semantic labels

A.2.6 Japanese

Amount labels: 72	T	E	D
perplexity	8.554	8.485	8.438
total amount	43957	5316	2706
GA	3.2955e+01%	3.2205e+01%	3.3444e+01%
WO	1.5224e+01%	1.4296e+01%	1.4228e+01%
NO	1.4810e+01%	1.6140e+01%	1.4893e+01%
NI	8.5083e+00%	8.7848e+00%	8.8322e+00%
DE	6.0900e+00%	5.9255e+00%	6.0237e+00%
TMP	5.4485e+00%	5.7374e+00%	5.5432e+00%
TO	4.6181e+00%	4.7028e+00%	5.0259e+00%
NoGap	3.8583e+00%	3.8375e+00%	3.7324e+00%
TO-IU	3.2782e+00%	3.4236e+00%	3.1042e+00%
KARA	1.5788e+00%	1.5989e+00%	1.6630e+00%
GA2	4.4362e-01%	5.0790e-01%	2.9564e-01%
HE	4.3907e-01%	3.3860e-01%	5.5432e-01%
TO-SHITE	3.8674e-01%	4.7028e-01%	4.8041e-01%
NI-TSUITE	4.0722e-01%	3.7622e-01%	3.3259e-01%
MADE	3.1394e-01%	2.8217e-01%	1.8477e-01%
NI-YOTTE	3.0029e-01%	2.2573e-01%	1.4782e-01%
NI-TAISHITE	1.5925e-01%	1.8811e-01%	1.8477e-01%
YORI	1.4560e-01%	9.4056e-02%	1.8477e-01%
NI-KANSHITE	1.3195e-01%	7.5245e-02%	1.8477e-01%
NI-MUKETE	9.7823e-02%	5.6433e-02%	1.8477e-01%
MADE-NI	9.5548e-02%	9.4056e-02%	7.3910e-02%
GA|WO	7.5073e-02%	7.5245e-02%	3.6955e-02%
WO-TSUUJITE	7.2798e-02%	1.8811e-02%	-
NI-TOTTE	7.0523e-02%	-	7.3910e-02%
WO-FUKUMETE	4.0949e-02%	3.7622e-02%	1.4782e-01%
NI-OITE	3.6399e-02%	1.5049e-01%	-
WO-MEGUTTE	3.4124e-02%	3.7622e-02%	1.8477e-01%
NI-KURABETE	4.3224e-02%	-	-
GA|TO	3.1849e-02%	3.7622e-02%	3.6955e-02%
GA|NI	2.7299e-02%	9.4056e-02%	-
NI-MOTOZUITE	2.5024e-02%	3.7622e-02%	1.1086e-01%
WO-NOZOITE	2.5024e-02%	-	-
NI-TOMONATTE	1.8200e-02%	-	-
NI-KARANDE	1.5925e-02%	-	-
NI-TSUZUITE	1.5925e-02%	-	-
NI-SHITE	1.5925e-02%	-	-
DE|GA	1.1375e-02%	-	3.6955e-02%
NI-KUWAETE	1.3650e-02%	-	-
TO-IU|WO	1.1375e-02%	-	-
NI-KAGITTE	1.1375e-02%	-	-
NI-KAWATTE	1.1375e-02%	-	-
GA|NO	1.1375e-02%	-	-
NO|GA	9.0998e-03%	-	-
NO|WO	9.0998e-03%	-	-
NoGap|WO	9.0998e-03%	-	-
DE|NI	4.5499e-03%	1.8811e-02%	-
NI-SOTTE	6.8249e-03%	-	-
NoGap|GA	6.8249e-03%	-	-
DE|NoGap	2.2750e-03%	-	3.6955e-02%
DE|TO	2.2750e-03%	1.8811e-02%	-
GA|TO-SHITE	-	1.8811e-02%	3.6955e-02%
NI-NARANDE	4.5499e-03%	-	-
MADE|KARA	2.2750e-03%	1.8811e-02%	-
TMP|TO	-	1.8811e-02%	-
NI|TMP	-	1.8811e-02%	-
TO-IU|TO	-	1.8811e-02%	-
DE|GA|WO	-	1.8811e-02%	-
YORI|NI	2.2750e-03%	-	-
DE|TO-IU	2.2750e-03%	-	-
NI-OITE|GA	2.2750e-03%	-	-
KARA|TMP	2.2750e-03%	-	-
NI|WO-FUKUMETE	2.2750e-03%	-	-
NO|NI-MUKETE	2.2750e-03%	-	-
NoGap|TMP	2.2750e-03%	-	-
NO|TO-IU	2.2750e-03%	-	-
NI-AWASETE	2.2750e-03%	-	-
TMP|WO	2.2750e-03%	-	-
GA|TMP	2.2750e-03%	-	-
GA|YORI	2.2750e-03%	-	-
GA|TO-IU	2.2750e-03%	-	-
NI|KARA	2.2750e-03%	-	-
NO|TMP	2.2750e-03%	-	-

Table A.6: Japanese semantic labels

A.3 Dependency relationship labels description on sub-trees

Originally some data-sets had multiple root nodes in a single sentence. Those nodes have a dependency relation "ROOT", with a non-existing head node. As the tree distance algorithm does not work with forest, an extra node was added to all sentences becoming the root of the tree and the head of the previous root nodes. This new root node has a dependency relationship "DepRel" with a non-existing head node. "Root" and "DepRel" are artificial labels, made to simplify the software. In some data sets those labels do not appear at all on the sub-trees, in other data sets those labels have an important frequency.

A.3.1 Catalan and Spanish

labels: 18	CA-T	CA-E	CA-D	ES-T	ES-E	ES-D
perplexity	7.1588e+00	7.2082e+00	7.2017e+00	7.0995e+00	7.0093e+00	7.0896e+00
total amount	121798	16276	16634	142878	16999	16677
suj	2.4454e+01%	2.4336e+01%	2.4252e+01%	2.4239e+01%	2.4213e+01%	2.3763e+01%
cc	1.9572e+01%	2.0122e+01%	1.9400e+01%	2.0996e+01%	2.1507e+01%	2.1335e+01%
cd	1.9318e+01%	1.8321e+01%	1.9653e+01%	1.9141e+01%	1.9519e+01%	1.9404e+01%
S	1.3151e+01%	1.3087e+01%	1.3593e+01%	1.3578e+01%	1.3336e+01%	1.3839e+01%
sentence	1.0470e+01%	1.0912e+01%	9.9134e+00%	9.5347e+00%	9.5770e+00%	9.3902e+00%
atr	4.0707e+00%	3.8830e+00%	3.8956e+00%	4.3247e+00%	4.1650e+00%	3.9336e+00%
creg	4.1355e+00%	4.4175e+00%	4.0399e+00%	3.0432e+00%	2.6766e+00%	3.0761e+00%
ci	1.4056e+00%	1.4684e+00%	1.6352e+00%	1.7301e+00%	1.5824e+00%	1.8229e+00%
cpred	1.4902e+00%	1.4500e+00%	1.5210e+00%	1.2948e+00%	1.2648e+00%	1.2652e+00%
cag	8.3253e-01%	8.8474e-01%	8.2361e-01%	7.8669e-01%	7.8828e-01%	8.2749e-01%
ao	7.3236e-01%	6.7584e-01%	9.1379e-01%	7.9858e-01%	7.3534e-01%	7.6752e-01%
sn	2.2414e-01%	2.6419e-01%	1.8035e-01%	3.3525e-01%	4.2944e-01%	3.4179e-01%
inc	1.3711e-01%	1.7818e-01%	1.7434e-01%	1.9177e-01%	1.9413e-01%	2.2786e-01%
grup.nom	4.9262e-03%	-	6.0118e-03%	2.7996e-03%	1.1765e-02%	-
s.a	8.2103e-04%	-	-	2.0997e-03%	-	-
sp	-	-	-	1.3998e-03%	-	5.9963e-03%
infinitiu	1.6421e-03%	-	-	-	-	-
mod	-	-	-	6.9990e-04%	-	-

Table A.7: DepRelation labels on the nodes of the sub trees for Catalan and Spanish

A.3.2 English

Amount DepRel labels: 65	T	E	D	ODD
perplexity	1.3485e+01	1.3505e+01	1.3895e+01	1.3920e+01
total amount	679902	39847	24206	5004
NMOD	1.8013e+01%	1.7718e+01%	1.7715e+01%	1.0791e+01%
SBJ	1.5967e+01%	1.6154e+01%	1.5744e+01%	1.7186e+01%
OBJ	1.4989e+01%	1.5055e+01%	1.3881e+01%	1.3849e+01%
PMOD	7.7357e+00%	7.4083e+00%	8.1096e+00%	6.0951e+00%
ROOT	7.6174e+00%	7.6091e+00%	7.5601e+00%	9.8921e+00%
ADV	6.0562e+00%	5.4985e+00%	5.7465e+00%	1.0572e+01%
VC	5.9512e+00%	6.4095e+00%	5.6184e+00%	6.0751e+00%
OPRD	2.9254e+00%	3.0391e+00%	3.2967e+00%	2.3181e+00%
TMP	2.7755e+00%	3.1169e+00%	2.9745e+00%	2.3981e+00%
IM	2.5580e+00%	2.6803e+00%	2.8464e+00%	2.1982e+00%
CONJ	2.0454e+00%	1.8069e+00%	2.2515e+00%	3.7970e+00%
LOC	1.9959e+00%	2.3264e+00%	2.4374e+00%	2.1183e+00%
SUB	1.9622e+00%	2.0177e+00%	2.0078e+00%	1.4988e+00%
APPO	1.7511e+00%	1.6513e+00%	1.7764e+00%	7.5939e-01%
COORD	1.6463e+00%	1.4932e+00%	1.9458e+00%	3.8969e+00%
PRD	1.2730e+00%	1.3627e+00%	1.5616e+00%	1.6587e+00%
DIR	8.4277e-01%	8.5075e-01%	5.8663e-01%	1.0991e+00%
PRP	7.3231e-01%	7.6041e-01%	7.0231e-01%	8.1934e-01%
MNR	7.1952e-01%	5.3204e-01%	5.4119e-01%	1.0592e+00%
AMOD	5.1552e-01%	5.2200e-01%	7.5188e-01%	4.5963e-01%
LGS	4.4065e-01%	4.9690e-01%	4.4617e-01%	1.9984e-02%
EXT	4.3183e-01%	4.2914e-01%	3.7594e-01%	-
DEP	3.7711e-01%	3.8146e-01%	4.5856e-01%	6.9944e-01%
HMOD	1.4237e-01%	1.6563e-01%	1.1154e-01%	5.9952e-02%
PRN	1.0531e-01%	9.2855e-02%	1.6525e-01%	5.9952e-02%
DTV	8.6630e-02%	7.0269e-02%	7.0231e-02%	1.9984e-02%
NAME	6.1038e-02%	4.7682e-02%	4.5443e-02%	-
PRT	5.1772e-02%	6.7759e-02%	3.7181e-02%	3.5971e-01%
EXTR	5.1772e-02%	5.5211e-02%	2.8918e-02%	1.9984e-02%
PUT	5.0890e-02%	4.2663e-02%	7.4362e-02%	1.9984e-02%
LOC-PRD	3.5152e-02%	8.5326e-02%	3.3050e-02%	1.9984e-02%
GAP-OBJ	1.4708e-02%	-	8.2624e-03%	5.9952e-02%
GAP-SBJ	1.3384e-02%	2.5096e-03%	1.2394e-02%	1.9984e-02%
DEP-GAP	8.5306e-03%	2.5096e-03%	4.1312e-03%	1.9984e-02%
BNF	7.0598e-03%	2.5096e-03%	8.2624e-03%	-
LOC-OPRD	5.5890e-03%	5.0192e-03%	2.0656e-02%	-
GAP-LGS	5.2949e-03%	7.5288e-03%	-	-
GAP-TMP	4.1182e-03%	1.2548e-02%	1.6525e-02%	-
GAP-PRD	4.7066e-03%	5.0192e-03%	-	-
DIR-GAP	3.9712e-03%	-	4.1312e-03%	-
EXT-GAP	3.3828e-03%	5.0192e-03%	-	-
P	3.5299e-03%	-	4.1312e-03%	-
GAP-LOC	3.0887e-03%	5.0192e-03%	4.1312e-03%	-
GAP-VC	2.2062e-03%	-	8.2624e-03%	-
TITLE	1.7650e-03%	-	-	-
PRD-TMP	1.7650e-03%	-	-	-
GAP-PMOD	1.3237e-03%	2.5096e-03%	-	-
VOC	1.3237e-03%	-	-	1.9984e-02%
PRD-PRP	1.1766e-03%	-	-	3.9968e-02%
ADV-GAP	1.4708e-03%	-	-	-
POSTHON	4.4124e-04%	2.5096e-03%	8.2624e-03%	-
DTV-GAP	8.8248e-04%	-	-	-
SUFFIX	7.3540e-04%	-	-	-
GAP-NMOD	5.8832e-04%	-	-	-
GAP-MNR	2.9416e-04%	-	-	1.9984e-02%
EXTR-GAP	2.9416e-04%	-	-	-
AMOD-GAP	2.9416e-04%	-	-	-
GAP-PRP	1.4708e-04%	-	-	-
GAP-LOC-PRD	1.4708e-04%	-	-	-
MNR-TMP	1.4708e-04%	-	-	-
DIR-OPRD	1.4708e-04%	-	-	-
MNR-PRD	1.4708e-04%	-	-	-
LOC-MNR	1.4708e-04%	-	-	-
GAP-OPRD	1.4708e-04%	-	-	-
GAP-SUB	1.4708e-04%	-	-	-

Table A.8: English DepRelation labels on the nodes of the sub trees

A.3.3 Chinese

Amount DepRel labels: 38	T	E	D
perplexity	9.8937e+00	9.8475e+00	9.9019e+00
total amount	377664	45231	29981
COMP	2.7796e+01%	2.8007e+01%	2.8702e+01%
ADV	1.7186e+01%	1.6977e+01%	1.6244e+01%
SBJ	1.6914e+01%	1.6597e+01%	1.6901e+01%
CJT	8.3227e+00%	8.6909e+00%	8.0985e+00%
ROOT	7.2557e+00%	7.0969e+00%	7.1979e+00%
TMP	3.7862e+00%	4.1476e+00%	4.1893e+00%
RELC	3.5095e+00%	3.4313e+00%	3.3655e+00%
CJTN	2.1718e+00%	2.2706e+00%	2.1747e+00%
PRD	2.0839e+00%	2.0031e+00%	1.8312e+00%
LOC	1.7621e+00%	1.8306e+00%	1.8645e+00%
AUX	1.7680e+00%	1.7488e+00%	1.6277e+00%
TPC	1.0536e+00%	8.9983e-01%	8.7722e-01%
MNR	9.9930e-01%	1.0126e+00%	1.1974e+00%
PRT	7.6576e-01%	7.3622e-01%	7.7382e-01%
DIR	7.5305e-01%	7.7602e-01%	6.6375e-01%
cCJTN	6.4369e-01%	6.8095e-01%	6.1372e-01%
PRP	4.8800e-01%	4.2007e-01%	5.0032e-01%
NMOD	4.4299e-01%	4.6871e-01%	6.1039e-01%
CND	4.3901e-01%	3.4269e-01%	4.0025e-01%
EXT	3.7335e-01%	4.9966e-01%	7.4380e-01%
OBJ	3.1271e-01%	2.6973e-01%	2.9352e-01%
APP	2.7273e-01%	3.1394e-01%	2.9018e-01%
UNK	2.4334e-01%	1.4813e-01%	1.6677e-01%
BNF	2.0547e-01%	2.0119e-01%	2.3348e-01%
FOC	1.6681e-01%	1.9456e-01%	1.1341e-01%
LGS	1.4907e-01%	1.3486e-01%	1.9346e-01%
IO	8.4996e-02%	8.1802e-02%	1.1674e-01%
OTHER	2.2507e-02%	8.8435e-03%	1.0006e-02%
VOC	1.8535e-02%	4.4217e-03%	6.6709e-03%
AMOD	2.6479e-03%	2.2109e-03%	-
DMOD	1.8535e-03%	2.2109e-03%	-
CJTN0	1.5887e-03%	-	-
CJTN1	1.0591e-03%	-	-
CJTN2	7.9436e-04%	-	-
CJTN3	5.2957e-04%	-	-
CJTN4	2.6479e-04%	-	-
CJTN5	2.6479e-04%	-	-
CJTN6	2.6479e-04%	-	-

Table A.9: Chinese DepRelation labels on the nodes of the sub trees

A.3.4 Czech

Amount DepRel labels: 48	T	E	D	.ODD
perplexity	9.1438e+00	9.3015e+00	9.2863e+00	8.8578e+00
total amount	912033	97928	122700	35790
Atr	3.7346e+01%	3.6463e+01%	3.6638e+01%	3.9276e+01%
Adv	1.2689e+01%	1.3199e+01%	1.3051e+01%	1.1520e+01%
Obj	1.0430e+01%	1.0654e+01%	1.0500e+01%	1.0699e+01%
Sb	1.0012e+01%	9.8440e+00%	9.9014e+00%	1.0117e+01%
AuxP	6.9143e+00%	6.7815e+00%	6.9764e+00%	6.2168e+00%
Atr_M	3.4868e+00%	3.5884e+00%	3.4808e+00%	2.5957e+00%
Pred	3.0500e+00%	3.0553e+00%	3.1524e+00%	2.8136e+00%
Coord	2.6414e+00%	2.6632e+00%	2.7253e+00%	2.2883e+00%
Pnom	1.8170e+00%	1.8646e+00%	1.8280e+00%	1.8273e+00%
Pred_M	1.8243e+00%	1.9749e+00%	1.8036e+00%	1.4138e+00%
Obj_M	1.6633e+00%	1.7758e+00%	1.5632e+00%	2.1039e+00%
Sb_M	1.4311e+00%	1.3786e+00%	1.6357e+00%	1.8050e+00%
AuxC	1.3133e+00%	1.3530e+00%	1.3301e+00%	2.3610e+00%
Adv_M	1.1400e+00%	1.1937e+00%	1.2160e+00%	1.0645e+00%
ExD	1.0915e+00%	1.0303e+00%	9.3073e-01%	8.1587e-01%
ExD_M	1.0527e+00%	1.0314e+00%	1.0196e+00%	4.6940e-01%
Apos	3.2137e-01%	3.4413e-01%	3.1296e-01%	6.1190e-01%
Coord_M	3.2598e-01%	3.5945e-01%	3.6838e-01%	1.7603e-01%
AuxZ	3.1359e-01%	2.8695e-01%	3.2681e-01%	3.6323e-01%
Atv	2.8091e-01%	3.2575e-01%	3.0644e-01%	2.5706e-01%
AtrAdv	2.1436e-01%	1.6747e-01%	2.3635e-01%	5.5882e-03%
Pnom_M	1.5383e-01%	1.6747e-01%	2.1760e-01%	1.8161e-01%
AtvV	9.1992e-02%	1.2356e-01%	9.8615e-02%	1.0897e-01%
AtrAtr	8.0260e-02%	5.2079e-02%	9.1280e-02%	-
AuxG	5.6467e-02%	5.4121e-02%	3.6675e-02%	6.1190e-01%
AdvAtr	4.2981e-02%	4.7994e-02%	3.5860e-02%	-
AtrAdv_M	3.2455e-02%	2.6550e-02%	3.2600e-02%	-
DepRel	3.0152e-02%	2.0423e-02%	2.4450e-02%	6.7058e-02%
Apos_M	3.0043e-02%	2.4508e-02%	2.9340e-02%	3.3529e-02%
AuxY	2.7960e-02%	2.9614e-02%	2.8525e-02%	7.2646e-02%
Atv_M	2.8398e-02%	2.9614e-02%	2.2820e-02%	2.5147e-02%
AuxT	1.9846e-02%	2.3487e-02%	1.8745e-02%	4.7499e-02%
AuxO	9.6488e-03%	1.7360e-02%	1.1410e-02%	-
AtrObj	7.0173e-03%	8.1693e-03%	8.1500e-03%	-
AtvV_M	5.5919e-03%	8.1693e-03%	1.1410e-02%	1.9559e-02%
AuxV	5.7015e-03%	6.1270e-03%	1.0595e-02%	1.3970e-02%
AdvAtr_M	5.3726e-03%	-	8.1500e-03%	-
ObjAtr	4.3858e-03%	6.1270e-03%	1.6300e-03%	-
AtrAtr_M	2.4122e-03%	1.4296e-02%	4.8900e-03%	-
AuxX	2.1929e-03%	4.0846e-03%	4.0750e-03%	8.3822e-03%
AtrObj_M	1.5350e-03%	-	-	-
AuxK	9.8681e-04%	2.0423e-03%	-	-
AuxV_M	6.5787e-04%	-	-	-
ObjAtr_M	4.3858e-04%	-	-	-
AuxZ_M	4.3858e-04%	-	-	-
AuxC_M	3.2894e-04%	-	-	-
AuxR	-	-	-	8.3822e-03%
AuxY_M	2.1929e-04%	-	-	-

Table A.10: Czech DepRelation labels on the nodes of the sub trees

A.3.5 German

Amount DepRel labels: 36	T	E	D	ODD
perplexity	7.5317e+00	7.3907e+00	7.3234e+00	7.6865e+00
total amount	60913	1910	2048	2282
SB	2.7075e+01%	2.7592e+01%	2.8076e+01%	2.4759e+01%
OC	2.0253e+01%	1.7906e+01%	1.6553e+01%	2.6161e+01%
ROOT	1.9328e+01%	2.0419e+01%	2.0850e+01%	1.5557e+01%
MO	1.0059e+01%	1.0314e+01%	1.0986e+01%	9.3777e+00%
OA	9.9814e+00%	1.0681e+01%	1.1328e+01%	8.8519e+00%
CJ	2.8303e+00%	2.8796e+00%	2.3438e+00%	4.7765e+00%
OP	2.6086e+00%	2.7749e+00%	2.6855e+00%	1.7090e+00%
RC	2.3591e+00%	2.5131e+00%	2.0508e+00%	2.4102e+00%
DA	1.6056e+00%	1.5183e+00%	1.3184e+00%	2.0158e+00%
RE	9.7024e-01%	7.3298e-01%	6.8359e-01%	1.2270e+00%
PD	9.3904e-01%	1.2042e+00%	1.6113e+00%	4.8203e-01%
CVC	4.4654e-01%	1.0471e-01%	9.7656e-02%	-
SBP	3.4804e-01%	3.6649e-01%	2.4414e-01%	3.9439e-01%
CC	2.2491e-01%	5.2356e-02%	1.4648e-01%	8.7642e-02%
MNR	2.0029e-01%	5.2356e-02%	1.4648e-01%	1.7528e-01%
PAR	1.6089e-01%	2.6178e-01%	1.4648e-01%	9.2025e-01%
NK	1.8879e-01%	1.0471e-01%	9.7656e-02%	2.1911e-01%
AG	1.6417e-01%	3.1414e-01%	1.9531e-01%	4.3821e-02%
RS	7.2234e-02%	5.2356e-02%	2.4414e-01%	-
OG	6.7309e-02%	5.2356e-02%	4.8828e-02%	1.3146e-01%
PG	2.6267e-02%	-	9.7656e-02%	-
APP	1.1492e-02%	-	-	3.0675e-01%
EP	1.3133e-02%	-	-	1.3146e-01%
DH	1.8059e-02%	-	-	-
CM	8.2084e-03%	-	4.8828e-02%	4.3821e-02%
CD	8.2084e-03%	5.2356e-02%	-	-
SVP	4.9251e-03%	5.2356e-02%	-	4.3821e-02%
NG	4.9251e-03%	-	-	4.3821e-02%
CP	3.2834e-03%	-	-	8.7642e-02%
PH	4.9251e-03%	-	-	-
DepRel	4.9251e-03%	-	-	-
JU	3.2834e-03%	-	-	-
OA2	3.2834e-03%	-	-	-
SP	1.6417e-03%	-	-	-
DM	1.6417e-03%	-	-	-
AC	-	-	-	4.3821e-02%

Table A.11: German DepRelation labels on the nodes of the sub trees

A.3.6 Japanese

Amount DepRel labels: 4	T	E	D
perplexity	1.1749e+00	1.1836e+00	1.1714e+00
total amount	146058	17511	9234
D	9.6441e+01%	9.6180e+01%	9.6383e+01%
P	3.3959e+00%	3.6891e+00%	3.5737e+00%
I	1.2392e-01%	1.0850e-01%	2.1659e-02%
ROOT	3.9026e-02%	2.2843e-02%	2.1659e-02%

Table A.12: Japanese DepRelation labels on the nodes of the sub trees

Appendix B

Software

This appendix explains the code of the tree-SRL system.

B.1 The code

The source code was written in C++, and it is composed of the following files:

file	lines of code	description
srl_2.cpp	804	This is the main executable file.
tree.h tree.cpp	526 3.200	They contain the code describing the basic tree structures.
TreeDistance.h TreeDistance.cpp	189 1.342	They contain the code for tree edit distance and similarity algorithm.
DepTreeCorpus.h DepTreeCorpus.cpp	153 3.317	They contain the code for loading and relabelling a file.
MAKEFILE		contains the script to compile the code.

Table B.1: Description of the source code files

The compilation was tested in Linux, and all source code can be downloaded with Apache license from: https://github.com/francoph/Tree-SRL.git

B.2 Sanity tests

This section describe a few tools and test done to verify the software works.

B.2.1 Self labelling

For any sub-tree the distance to itself should be the minimal possible distance (zero). Therefore, if a single data set file is used to label itself with k = 1, wrong predictions can only be made when multiple neighbours are counted.

In all the experiments where a data set was used to label itself, no single case was found of a sample in which a single nearest neighbour leads to a wrong prediction.

B.2.2 Nearest cut and k-NN

An artificial data set was created with eleven samples for labelling just one sub-tree with a single argument in which all samples were usable, with a first equivalence class containing one sample and the second equivalence class containing another ten samples.

When the system was asked to use a k value equal or less than five, the panel size was reduced to one in order to fit the first equivalence class, but for any value of k equal or over six, the panel size changed to eleven in order to fit the second equivalence class.

B.2.3 Normalised distance and similarity

Theoretically if a distance and similarity measure are A-dual and N-duals then the system will produce the same results using any of them. Normalised distance A-dual to normalised similarity accomplishes this property and as expected the results for both settings are identical.

B.2.4 Graphical inspections

A few modifications in the source code can make the system search two specified sub-trees, produce an alignment to each other and display both sub-trees, and their alignment.

B.3 User manual

This section describes how to use the implementation of the system.

The system should be executed with the following parameters:

compusery

- -training + trainingFile

  This is a required parameter where the argument is the name of the training file.

- -testing + testing File

  This is a required parameter where the argument is the name of the testing file.

- -tag Tagforexperiment

  This is an optional parameter but it is highly recommended to use it. Its argument is a string. This argument will be added to the file name of the output files and it is used to identify them and differentiate training and testing data set used on the experiment. If it is missing the system will produce a random label.

tunning

- -allArgumentsEqual

  This is an optional parameter without arguments. It makes replaces all semantic labels into 'argument', and it is used to do experiments on argument identification.

- -identificationTask

  This is an optional parameter without arguments. It makes all non arguments nodes in a sentence become an argument with a semantic label '?NoPrediction' to the predicate node, for each predicate. This is used to perform an identification argument task.

- -deleteDepRel

  This is an optional parameter without arguments. It replaces the labels for all dependency relationships into 'hidden'. This was used to calculate the Leveinstain distance without using dependency relationships.

- -atomicMeasure

  This is an optional parameter. It allows the user to select the atomic measure for the tree edit distance module, where the options are:

  - binary
  - ternary
  - hamming
  - shape
  - ALPHATERNARY
  - ALPHAHAMMING

  All similarity versions are made by adding a prefix 's' to the argument label.
  The default value if binary.

- -delta

  This parameter is only compulsory when using ALPHATERNARY or ALPHAHAMMING atomic measures.
  If it is not specified by default the value is one. The argument should be a real number.

- -insertCost double

  This is an optional parameter with one argument, it modifies the insertion/deletion cost in the case of distance settings.
  In the case of similarity settings insertion/deletion cost will be zero, but it will modify the swapping scores which are defined as two times the distance deletion/insertion cost minus the distance swapping cost.
  By default the value of the argument is one. The argument should be a real number.

- -k number

  This is an optional parameter with one argument which modifies the preferment k value of the k-NN module, by default its argument value is one. The argument should be an integer positive number.

- -normalise $Y|N$
  This is an optional parameter, with one argument, which allows the user to normalise the distance or similarity scores. By default the system does not normalise. The argument can be 'Y' for normalising or 'N' for not normalising.

- -multipleArguments $Y|N$
  This is an optional parameter with one argument which decides if the system use sub-trees with multiple arguments or sub-trees with a single argument.
  By default the system uses sub-trees with multiple arguments.

- -maxSentences number
  This is an optional parameter with one argument which allows the user to limit the size of the data sets used to the specified amount of first sentences. The argument should be an integer positive number.

- -leveinstan
  This is an optional parameter without arguments. It will load the corpus as strings in word order and not as trees.

- -1N
  This is an optional parameter without arguments. It will disable the k-NN module using each sample as it is an equivalence class.

- -predMatchCost double
  This is an optional parameter with one argument.
  The argument specifies the frame mismatch cost, which will add extra cost to the swamping costs of predicate nodes which belongs to different frame or swaps between predicate nodes and non-predicate nodes. The argument should be a real number, by default its zero.

statistics

- -getSingleRootedSentences
  This counts how many roots have the sentences of the training data set.

- -stats

  This is an optional parameter. It shows statistics of the training data set.

- -nullPrediction $Y|N$

  This is an optional parameter. It allows the user to use the null votes on the k-NN module as real votes, which if they win, the system will predict a non-label. By default it is deactivated.

- -compareDataSets

  This is an optional parameter with unlimited arguments. It can be called only without any other parameters. The arguments will be considered as files to compare with each other. In this case the system will generate a table in latex format comparing the datasets, the tables in appendix A were generated with this function.

orders

- -predict
  This is an optional parameter without arguments. It gives the order to process the experiment, it need to have the neighbourhood distances pre-calculated.

- -generateTables
  This is an optional parameter without arguments. It gives the order to pre-calculate the neighbourhood distances, which will be used on the k-NN module for making the predictions. It will generate a file *.tre .

B.4 Computational Cost Analysis

This section describes the computational cost for labelling the testing data sets.

For classifying a single sample, it is required to know the distance to all samples on the training data set, or at least the distance of the nearest ones, which usually requires knowing all the distances. Therefore, the cost of labelling a test-file will be quadratic in proportion to the size of the testing data set and the evaluation data set on the amount of predicates.

As the experiments have to be carried out with different values for the parameter k, the software is designed for saving the table of distances in a temporal file, and in a second phase, use those temporal files for labelling the evaluation data set.

As the total amount of space required to save all files generated for each

Table B.2: k-NN cost for the different languages

The estimated size in Gigabytes (GB) is made by estimating 12 Bytes per comparison, different data sets change four orders of magnitude in computational cost.
Czech and English training data sets were reduced in order to make them easy to process, but the reduction was about a quarter of the original data set.

language	Predicates training set	Predicates testing set	Millions of comparisons	Estimated size in GB	GB saving top 1,000
Czech	414,237	44,585	18,468.76	206.4	44.59
English	179,014	10,498	1,879.29	21.0	10.50
Chinese	102,813	12,282	1,262.75	14.1	12.28
Spanish	43,824	5,175	226.79	2.5	5.18
Catalan	37,431	5,001	187.19	2.1	5.00
Japanese	25,712	3,111	79.99	0.9	3.11
German	17,400	550	9.57	0.1	0.55

different setting of tree distance is larger than our resources, only the top thousand nearest or similar samples are saved in the distance file, it reduces the hard disk requirements but not the computational time.