Analyzing, Enhancing, Optimizing and Applying Dependency Analysis
Ph. D. Thesis

Miguel Ballesteros Martínez

under the supervision of:
Virginia Francisco. Ph. D
Pablo Gervás. Ph. D

Outline

Introducción

- Los analizadores de dependencias estadísticos han sido mejorados en gran medida durante los últimos años.
- En esta tesis, los utilizamos con distintos objetivos:
  - Estudio de los analizadores.
  - Mejora de los analizadores.
  - Optimización de los analizadores.
  - Aplicación de estructuras de dependencias.

Introduction

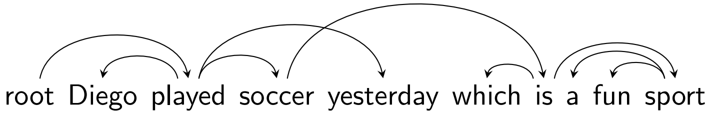
Dependency Parsing

- Syntactic parsing of natural language.
- Dependency-based syntactic representations:
  - Long tradition in descriptive and theoretical linguistics.
  - Increasingly popular in computational linguistics.

Introduction
Dependency Structures

In a formal fashion, a **dependency graph** for a Sentence, S = w_1,...,w_n is a directed graph G = (V,A), where:

- V = $\{1, ...n\}$ is the set of **nodes**, representing tokens.
- A $\subseteq$ V x V is the set of **arcs**, representing dependencies.
- An arc i $\rightarrow$ j is a dependency with head w_i and dependent w_j.
- An arc i $\rightarrow$ j may be labeled with a dependency type contained in the set of possible dependencies.



Introduction

Dependency Structures

- The dependency graphs are normally assumed to satisfy certain formal constraints:
  - Single head constraint.
  - Aciclicity constraint.
- Many annotation schemes further require that the graph should be a tree, with a unique root token.

Introduction

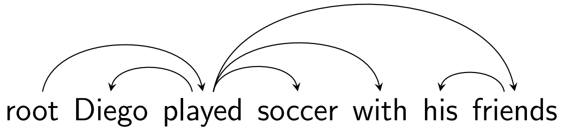
Dependency Structures. Projectivity and Non-Projectivity

A dependency tree T is projective iff

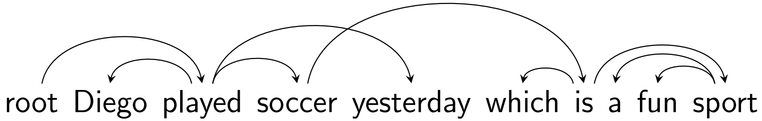
- for every arc wi $\rightarrow$ wj and every node wk between wi and wj in the linear order, there is a (directed) path from wi to wk.
- "No crossing edges".
- Non-projective structures are needed to account for
  - Long distance dependencies.
  - free word order.

Introduction

Basic Assumptions. Projectivity and Non-Projectivity



Projective Tree

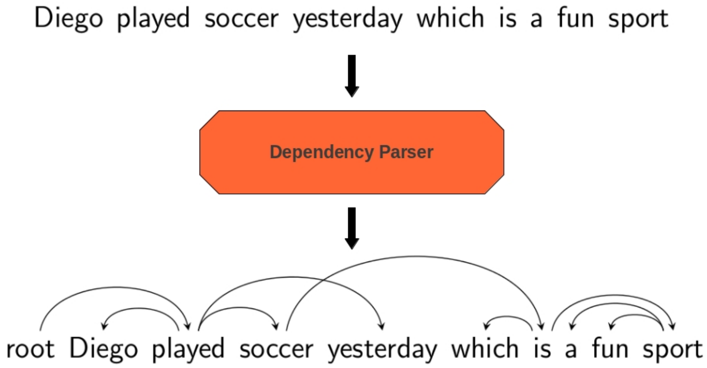


Non-Projective Tree

Introduction

Dependency Parsing

We have an input sentence, we produce a dependency tree for the given sentence.

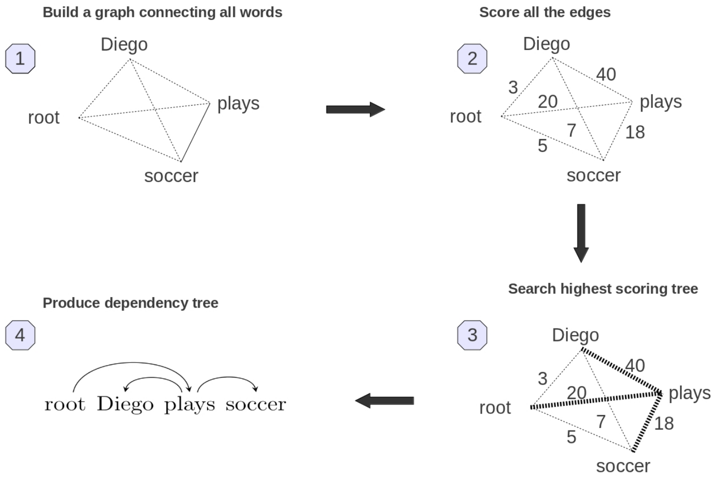

Diego played soccer yesterday which is a fun sport
Dependency Parser
root Diego played soccer yesterday which is a fun sport

Approaches:

- Rule-based: (Late 90's and early 2000)
  - Grammar-based parsing
    - In this thesis we used Minipar (Lin et al.)
- Data-Driven: (2005 and ongoing)
  - Graph-based parsing
    - In this thesis we used MSTParser (McDonald et al.)
  - Transition-based parsing.
    - In this thesis we mainly used **MaltParser** (Nivre et al.)

Introduction

Graph-Based Dependency Parsing



Introduction

Transition-Based Dependency Parsing

Reduces the problem of parsing a sentence to the problem of finding an optimal path through an abstract transition system, or state machine.

- It processes the sentence from left to right in a deterministic way.
- It has two data structures: the stack and the buffer.
- In each step the parsing algorithm have to select one operation: left-arc, right-arc, shift (reduce) or swap.
- A classifier (linear SVM) is used to select the transition.

Introduction

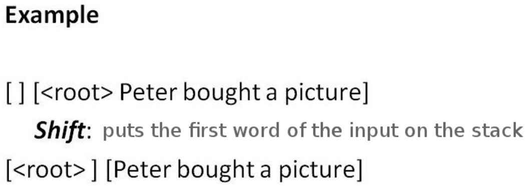
Transition-Based Dependency Parsing

Example

[ ] [<root> Peter bought a picture]

Introduction

Transition-Based Dependency Parsing



Example

[ ] [<root> Peter bought a picture]

Shift: puts the first word of the input on the stack

[<root>] [Peter bought a picture]

Example

[ ] [<root> Peter bought a picture]

Shift: puts the first word of the input on the stack

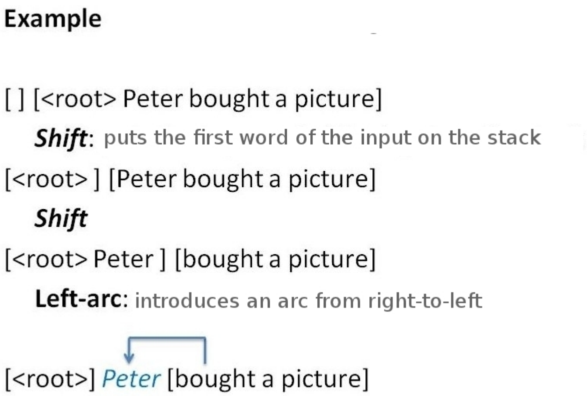
[<root>] [Peter bought a picture]

Shift

[<root> Peter] [bought a picture]

Introduction

Transition-Based Dependency Parsing



Example

[ ] [<root> Peter bought a picture]

Shift: puts the first word of the input on the stack

[<root>] [Peter bought a picture]

Shift

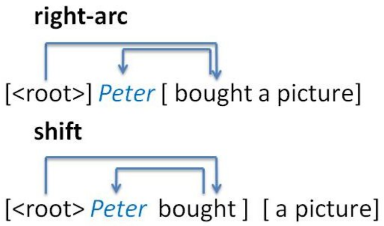
[<root> Peter] [bought a picture]

Left-arc: introduces an arc from right-to-left

[<root>] *Peter* [bought a picture]

Introduction

Transition-Based Dependency Parsing


right-arc
[<root>] *Peter* [bought a picture]
shift
[<root> *Peter* bought] [a picture]

...

Introduction
MaltParser

MaltParser is a transition-based dependency parser generator.

- This thesis is inherently related to MaltParser.
- Contains four different parsing algorithms.
- It uses support vector machines (either LIBSVM or LIBLINEAR).

Introduction
MaltParser-Algorithms

1. **Nivre's algorithms**: include two algorithms, the **arc-eager** and **arc-standard**.
2. **Covington's algorithms**: include the projective and non-projective versions. Similar to Nivre arc-eager, concerning parsing order.
3. **Stack algorithms**: they include the projective and non-projective versions. Similar to Nivre arc-standard parsing algorithms.
4. **Multiplanar parsers**: include two algorithms; the Planar parser and the 2-Planar parser. Similar to Nivre arc-eager.

- All of the projective parsers can be run with pseudo-projective parsing (Nivre, 2005).

Introduction
Parsing Evaluation

In the data-driven setup we have the following:

- A training corpus containing a (big) set of examples.
- A test corpus containing a smaller set of examples that serves for evaluation.
- There are corpora for more than 25 languages annotated in a standard format: CoNLL data format.

Evaluation

There are several ways of evaluating the parsers:

- Labelled and unlabelled attachment scores. UAS and LAS.
- Label accuracy. LA
- Labelled and unlabelled complete-match scores.

Introduction
Goals

In this thesis we studied dependency parsing from several perspectives:

1. Studying the problem focusing on the corpora size and sentence length.
2. Improving the outcomes of the parsers.
3. Optimizing the parsers or so called, machine learning models.
4. Applying dependency structures to solve NLP problems.

Analyzing Dependency Analysis

- Studies about the homogeneity of the dependency parsing problem focusing on training corpora size and sentence length.
    - Are the existing training corpora unnecesarily large?
    - Are the longer sentences more useful for training than the shorter ones?

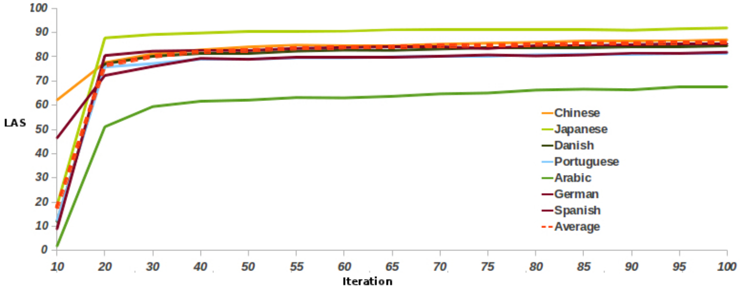
All of these experiments were addressed in the same direction: finding corpora limitations and the behavior of the parsers.

Analyzing Dependency Analysis

Training Corpora Size and Sentence Length

Question: are the training corpora unnecesarily large?

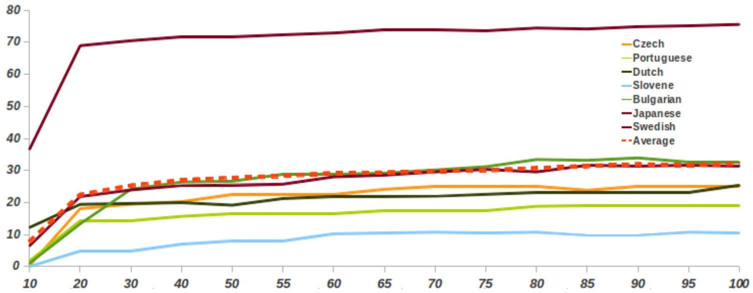
- We fragmented the corpora in small pieces and we trained MaltParser in default settings over them getting different models.



Analyzing Dependency Analysis

Training Corpora Size and Sentence Length

Question: are the training corpora unnecesarily large?



Analyzing Dependency Analysis

Training Corpora Size and Sentence Length

Question: are the training corpora unnecesarily large?

- The corpora are not unnecesarily large, we always get improvements by adding more words. *However...*
- After a certain limit, the improvement seems not significant.
- Building a corpus of a big size is a very big effort.
- It is good to know that we can get results in the same range of accuracy with reduced corpora. Models trained over a small corpus are:
  - Faster in training time.
  - Faster in parsing time.
  - Less noise in the accuracy if the annotation provided is not very consistent.

Analyzing Dependency Analysis
Training Corpora Size and Sentence Length

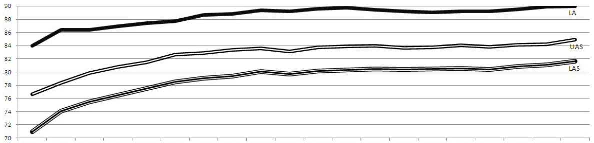
Question: are the long sentences more useful for training than the short sentences?

- The experimental set-up was similar to the previous one.
- But, in this case, we train iterative models including firstly longer sentences.

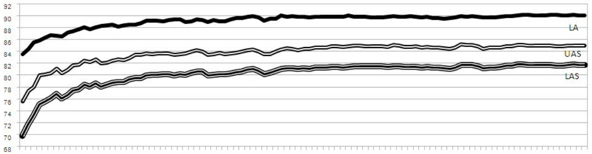
Analyzing Dependency Analysis

Training Corpora Size and Sentence Length

Train iterative models including firstly longer sentences.



Incremental corpora without taking into account the sentence length.



Incremental corpora prioritizing the longer sentences.

Analyzing Dependency Analysis

Conclusions.

- We can conclude that long sentences are very useful for training.
  - Even with similar sizes of the training corpus.
- We them emphasize the inclusion of longer sentences in the training corpora instead of prioritizing the final size.

Enhancing Dependency Analysis

- Feasibility study about output parsing modifications.
- Study about the root position during parsing and training time.

Enhancing Dependency Analysis

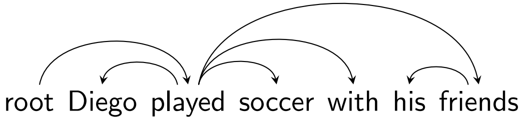
Feasibility study about output parsing modifications

- We studied the feasibility of a system that modifies the annotation in a post-processing step.
- We focused in function words, such as prepositions, the conjunction and the nexus. In the Spanish case.
- The results were very promising, however when we wanted to try the real case, the results were just not too bad.
- These words were the root of subtrees and this is one of the things that lead us to study the root position with the idea of enhancing the performance.

Enhancing Dependency Analysis

Study about the root position during parsing and training time

- Dependency trees used in syntactic parsing often include a root node representing a dummy word **prefixed** to the sentence.
- It is tacitly assumed to have no impact on empirical results.
- However, it is well known that in a multi-clause sentence is very likely to find several post-subordinate clauses but it is extremely uncommon to have more than one pre-subordinate clause.



Enhancing Dependency Analysis

Study about the root position during parsing and training time

So far, researchers assummed that the root position during parsing and training time does not affect the accuracy. **Is it really true?**

- The **Nivre arc-eager** parsing algorithm is greedy generating left attachments.
- However, other parsing orders, such as the one represented by **Nivre arc-standard** parsing algorithm is more lazy about that.

Hypothesis

The root position really matters, and it affects the accuracy in some parsing algorithms, such as the Nivre arc-eager.

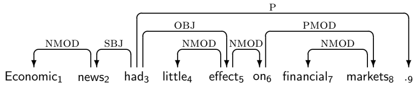
- Let's find out...

Enhancing Dependency Analysis

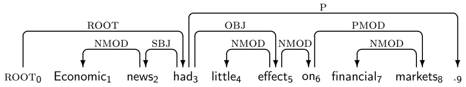
Study about the root position during parsing and training time

Three scenarios:

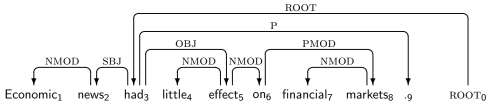
- No root.
- Root to the left.
- Root to the right.



P
OBJ
PMOD
NMOD
SBJ
NMOD
NMOD
NMOD
Economic1 news2 had3 little4 effect5 on6 financial7 markets8 .9



P
ROOT
OBJ
PMOD
NMOD
SBJ
NMOD
NMOD
NMOD
ROOT0 Economic1 news2 had3 little4 effect5 on6 financial7 markets8 .9



ROOT
P
OBJ
PMOD
NMOD
SBJ
NMOD
NMOD
NMOD
Economic1 news2 had3 little4 effect5 on6 financial7 markets8 .9 ROOT0

Enhancing Dependency Analysis

Study about the root position during parsing and training time

- We modified the corpora in order to have the constraints.
- We run:
  - Nivre arc-eager algorithm.
  - Nivre arc-standard algorithm
  - Arc-Factored Spanning Tree Parsing: MSTParser.
- For MaltParser algorithms (arc-eager and arc-standard) we run the parser:
  - Without root during parsing. Just use the root of the corpora.
  - With pseudo-projective parsing in order to reduce the noise produced by the non-projective arcs.
- We had to modify the MSTParser source code in order to emulate the same scenarios.

Enhancing Dependency Analysis

Study about the root position during parsing and training time

		Nivre arc-eager		Nivre arc-standard		MSTParser	
Language	Type	LAS	UAS	LAS	UAS	LAS	UAS
Arabic	None	*60.00*	**75.17**	*60.48*	77.29	**66.73**	**78.96**
	Left	63.63	74.57	64.93	77.09	66.41	78.32
	Right	**64.15**	74.97	**65.29**	**77.31**	66.41	78.32
Czech	None	*68.30*	81.14	*68.36*	81.96	76.70	85.98
	Left	72.98	79.96	**74.88**	**82.52**	77.04	86.34
	Right	**73.96**	**81.16**	74.28	81.78	**77.68**	**86.70**
Danish	None	82.36	87.88	81.64	**87.86**	83.39	89.46
	Left	80.60	86.59	**81.66**	**87.86**	**83.97**	**89.84**
	Right	**82.38**	**87.94**	81.52	87.74	83.43	89.42
German	None	83.85	86.64	84.31	87.22	85.64	89.54
	Left	83.29	86.08	**84.37**	**87.24**	**85.74**	**89.66**
	Right	**83.93**	**86.72**	84.35	87.22	85.34	89.50
Portuguese	None	*79.12*	88.60	*78.32*	87.78	84.87	89.74
	Left	83.77	88.36	83.45	87.82	**85.19**	**90.26**
	Right	**84.17**	**88.62**	**83.47**	**87.80**	84.89	89.26

Enhancing Dependency Analysis

Study about the root position during parsing and training time

Conclusions

- For certain parsing models, the existence and placement of the dummy root node is in fact a parameter worth tuning for best performance.
- The dummy root node may be an underestimated source of variation and a variable that needs to be controlled for experimental evaluations.
- We believe that it may change the way the researchers think about it.

- We (Joakim Nivre and I) submitted a complete paper about this experiment in the CL journal.

Many data-driven systems require careful tuning, which may require specialized knowledge of the system and the task.

What do we need to optimize in a transition-based parser?

- Parsing Algorithm.
- Feature Model.
- Machine Learning Algorithm.

In order to do this, we first need to analyze the data.

Optimizing Dependency Analysis

MaltOptimizer

- To do that, we developed a tool for parsing optimization.
- We called it **MaltOptimizer**, from **Malt**Parser **Optimiz**ation.
- http://nil.fdi.ucm.es/maltoptimizer
- Based on three different phases:
  1. Data Characteristics.
  2. Parsing Algorithm.
  3. **Feature Selection** (and Machine Learning Algorithm).

Optimizing Dependency Analysis

Basic Assumptions

MaltOptimizer has the following basic assumptions.

- Input: CoNLL data format (Dependency treebank).
- Evaluation metric: Labelled Attachment Score (may accept Unlabelled).
- Machine Learner: LIBLINEAR / multiclass SVM.
- Validation Methods:
  - Train development - test split (80% - 20%).
  - 5-Fold cross validation.

Optimizing Dependency Analysis
CoNLL Data format

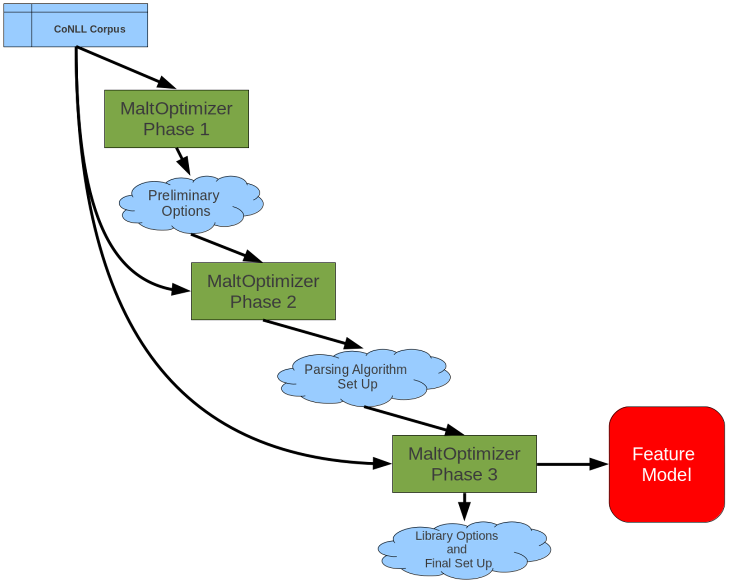
The following attributes are available in the CoNLL format assumed by the parser.

1. FORM: Word form.
2. LEMMA: Lemma.
3. CPOSTAG: Coarse-grained part-of-speech tag.
4. POSTAG: Fine-grained part-of-speech tag.
5. FEATS: List of morphosyntactic features (e.g., case, number, tense, etc.).
6. DEPREL: Dependency relation to head.

Note

- LEMMA and FEATS are not available in all data sets.
- CPOSTAG and POSTAG tags are sometimes identical.

Optimizing Dependency Analysis



Optimizing Dependency Analysis

Phase 1: Data Analysis

In the data analysis phase, MaltOptimizer gathers the following:

1. Number of words/sentences.
2. Percentage of non-projective arcs/trees.
3. Existence of “covered roots”.
4. Frequency of labels used for tokens with HEAD $= 0$.
5. Non-empty feature values in the LEMMA and FEATS columns.
6. Identity of feature values in the CPOSTAG and POSTAG columns.

Optimizing Dependency Analysis

Phase 1: Data Analysis - Non-Projectivity

- There are several methods to deal with non-projective structures in transition-based dependency parsing.
- We must therefore select the algorithm carefully.

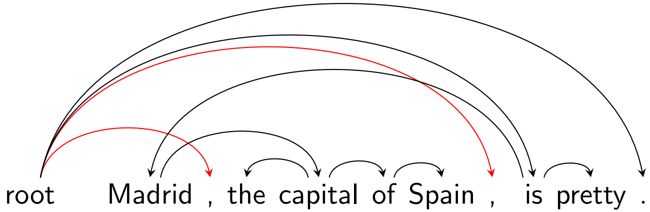
Conclusion

It is necessary to acknowledge the proportion of non-projective trees in a corpus in order to achieve an optimal performance.

Optimizing Dependency Analysis

Phase 1: Data Analysis - Covered Roots

- It is a normal situation when we have dangling punctuation: punctuation tokens with HEAD = 0 (Czech, Danish or Greek).
- In the general case, this case occurs when we have a root node (HEAD = 0) covered by an arc that is not connected to the root.



Optimizing Dependency Analysis

Phase 1: Data Analysis - Covered Roots

- MaltParser contains a specific option to handle covered roots.
- Running the parser without knowing the percentage of covered roots would lead to an incontrollable number of spurious non-projectivities.

Conclusion

It is necessary to acknowledge the proportion of covered roots in a corpus.

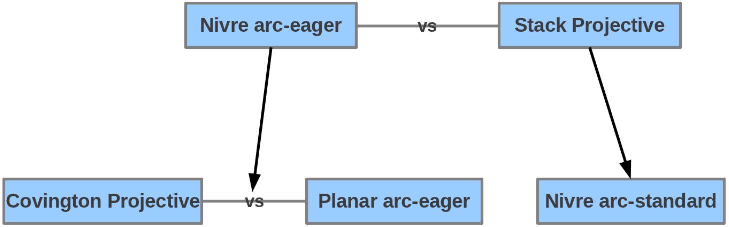
Optimizing Dependency Analysis

Phase 2: Parsing Algorithm

- MaltParser implements three groups of transition-based parsing algorithms:
  1. Nivre's algorithms.
  2. Covington's algorithms.
  3. Stack algorithms.
- Both the Covington group and the Stack group contain algorithms that can handle non-projective dependency trees.
- Any projective algorithm can be combined with pseudo-projective parsing.

Optimizing Dependency Analysis

Phase 2: Parsing Algorithm

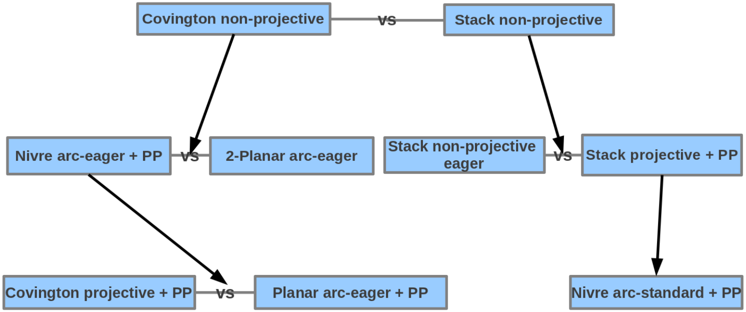


Projective situation.

The system traverses this decision tree if the percentage of non-projective trees is not too high (>15%).

Optimizing Dependency Analysis

Phase 2: Parsing Algorithm - Non-Projective situation



Non-Projective situation.

The system traverses this decision tree if the percentage of non-projective trees is not zero.

Optimizing Dependency Analysis

Phase 3: Automatic Feature Selection

- The choice of features to build NLP based on machine learning applications is an unavoidable task.
- In automatic feature selection we select the features that perform better for a single task.
- Nowadays, it is a matter of linguistic and task expertise.

Conclusion

Selecting the right features is an important task for optimization.

Optimizing Dependency Analysis

Phase 3: Automatic Feature Selection

- MaltParser uses history-based feature models.
- It uses
  - features over the partially built dependency structure.
  - features of the annotated input string.

Optimizing Dependency Analysis
Phase 3: Default Feature Models in MaltParser

1. A wide window of POSTAG features over the stack and buffer (typically of length 6).
2. A narrower window of FORM features over the stack and buffer (typically of length 3).
3. A small set of DEPREL features over dependents (and heads) of the most central tokens on the stack and in the buffer (typically of size 4).
4. A small set of combinations of the above features, in particular POSTAG n-grams and pairs of POSTAG and FORM features.

Optimizing Dependency Analysis
Phase 3: Algorithm Steps

1. Tune the window of POSTAG n-grams over the stack and the buffer.
2. Tune the window of FORM features over the stack and the buffer.
3. Tune DEPREL and POSTAG features over the partially built dependency tree.
4. Add POSTAG and FORM features over the input string.
5. Add CPOSTAG, FEATS, and LEMMA features if available.
6. Add conjunctions of POSTAG and FORM features.

Optimizing Dependency Analysis

Phase 3- Feature Selection Greedy Algorithm

- Traverses all the 6 steps adding one feature at a time and keeping the feature set that provides the best result so far.
- First, backward selection.
- Then, forward selection of new features.

Optimizing Dependency Analysis

Experiments

- Data sets:
  - CoNLL 2006: 13 languages.
  - CoNLL 2007: 10 languages.
- Optimization in 3 phases, using either dev-test split or cross validation, depending on the size of the corresponding treebank.
- We also report results compared to manual optimization, the one provided in the Shared Tasks (2006-2007).

During the presentation I will only show results from the CoNLL 2007 Shared Task, you can find it in the thesis.

Optimizing Dependency Analysis

Results-CoNLL-2007 Shared Task

CoNLL-2007 Shared Task							
Language	**Default**	**Phase 1**	**Phase 2**	**Phase 3**	**Diff**	**Test-MO**	**Test-MP**
Arabic	67.71	67.75	67.75	70.77	3.06	73.22	**74.75**
Basque*	67.69	67.83	68.29	75.05	7.36	72.19	**74.99**
Catalan	83.07	83.07	83.13	84.89	1.82	85.87	**87.74**
Chinese	84.04	84.04	85.03	86.21	2.17	82.58	**83.51**
Czech	70.25	70.51	72.49	77.71	7.46	**78.03**	77.22
English	83.84	83.84	85.34	86.61	2.77	85.17	85.81
Greek*	71.01	71.09	72.41	75.12	4.11	**74.50**	74.21
Hungarian	66.42	66.42	68.21	76.53	10.11	77.17	**78.09**
Italian*	79.07	79.07	79.45	81.53	2.46	**82.79**	82.48
Turkish*	67.45	68.38	70.67	76.91	9.46	78.93	**79.24**

Optimizing Dependency Analysis
Conclusions

- Experiments show consistent improvements, varies from...
    - About 2 percentage points for some languages, such as Catalan or Chinese.
    - Up to 7–10 points for Basque, Czech, Hungarian and Turkish.
- Some data sets include rich linguistic annotation. Greatest improvements from phase 3.
- We also see significant improvement from phase 2, due to non-projective dependencies.

We have received a lot of interest.

Applying Dependency Analysis

Here we present some works that we did in parallel with the intention of showing the potential usefulness of dependency parsing.

- Text Simplification.
- Inferring the Scope of Negation.

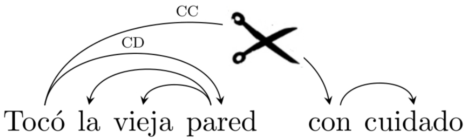
Applying Dependency Analysis

Text Simplification

- Can we simplify texts by using dependency parsers?
- Text simplification is very useful for education
- We developed a rule-based system.
- We evaluated with a survey for children and adults.

Applying Dependency Analysis

Text Simplification



- Our system prunes the dependency trees according to the dependency labels.
- It removes complementary information about an action.
- The results show improvements in the understanding.
- The main conclusion is that by using dependency parsing the resulting sentences are grammatically correct and easier to read.

Applying Dependency Analysis

Inferring the Scope of Negation

- Every text contains information that includes uncertainty, deniability or speculation.
- It is important to distinguish between negative statements and factual ones.
- Chapman et al. (2002) proved that in a search for *fracture* in a radiology reports database, 95 to 99 percent of the reports returned would state *"no signs of fracture"* or words to that effect.
- In information retrieval systems, it seems necessary to acknowledge whether words have been negated or not.

Applying Dependency Analysis
Inferring the Scope of Negation

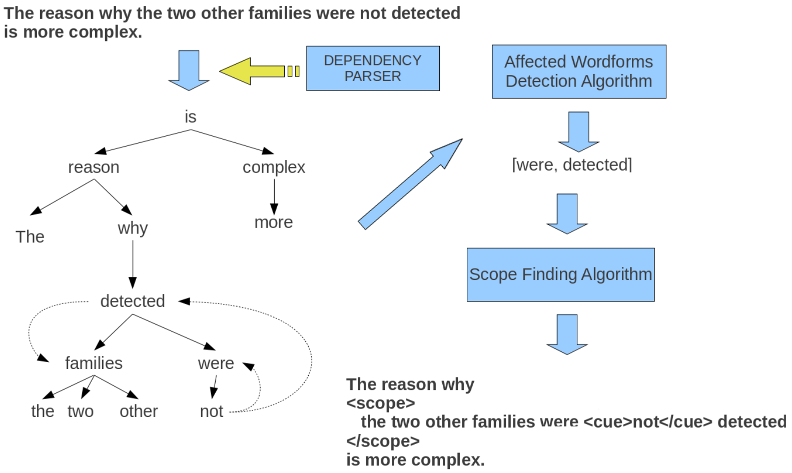
- The dependency parsers return very good results, if we consider global accuracy. More than 85% LAS for English parsing.

Our goal was to build a system to promote the inferring of the Scope of Negation for English sentences.

- Can we use dependency parsing as a tool to do that?
- We used the Minipar parser as a source.
- The system is **rule-based**.

Applying Dependency Analysis

Inferring the Scope of Negation



The reason why
<scope>
the two other families were <cue>not</cue> detected
</scope>
is more complex.

Applying Dependency Analysis

Inferring the Scope of Negation

Evaluation

We carried out the evaluation over the Bioscope corpus.

- It is divided in three different collections.
- Linear annotation of the scopes.

Collection	Precision	Recall	F1	PCS	PCNC
Papers	73.49%	80.70%	76.93%	56.43%	91.15%
Abstracts	84.92%	84.03%	84.48%	68.92%	95.56%
Clinical	95.83%	90.58%	93.13%	89.06%	94.82%

According to the systems that evaluated with the same corpus our results are very competitive.

Participation in the *SEM Shared Task

- The *SEM Shared task was a competitive task of systems that infer the scope of negations.
- We presented our system based on Bioscope modifying the behavior.
- All of the systems were developed for the Shared Task, however, our system was not.
- We ranked fourth over five systems.
- The results were competitive measuring the words inside the scopes.

Conclusions

- From the Analysis branch of the thesis:
    - We concluded that the training corpora can be definitely optimized in two ways.
    - Prioritize long sentences.
    - Do not Prioritize the final size of the corpus.
- From the Enhancing branch of the thesis:
    - We showed two very interesting experiments in which the root of trees and subtrees are addressed.
    - The second one may change how the researchers think about the root of trees which is something essential.

Conclusions

- From the Optimizing branch of the thesis:
  - We produced a very good system that is capable of providing an optimized version of parsing models.
  - Moreover, these models may even produce better results that models manually optimized by experts.
- From the Applying branch of the thesis:
  - We demonstrated that the dependency structures are very useful to solve NLP problems.

Future Work

There may be several future directions:

- Explore new ways of enhancing the accuracy of the parsers focusing in the trees.
- Explore new automatic feature selection algorithms in the optimization branch.
- Apply dependency parsers to new NLP problems.
- We could also try to find ways of speeding up the process and produce parsers for the real life.

Publications

Journal Papers

1. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2010. Improving Parsing Accuracy for Spanish using Maltparser. Journal of the Spanish Society for Natural Language Processing (SEPLN) number 44.
2. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2012. Analyzing the CoNLL-X Shared Task from a Sentence Accuracy Perspective. Journal of the Spanish Society for Natural Language Processing (SEPLN) number 48.
3. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2012. Are the existing training corpora unnecessarily large?. Journal of the Spanish Society for Natural Language Processing (SEPLN) number 48.
4. **Miguel Ballesteros**, Carlos Gómez-Rodríguez and Joakim Nivre. 2012. Optimizing Planar and 2-Planar Parsers with MaltOptimizer. Journal of the Spanish Society for Natural Language Processing (SEPLN) number 49.

- (Submitted) **Miguel Ballesteros** and Joakim Nivre. 2012-2013?. Getting to the Root of the Problem. Computational Linguistics Journal. JCR.

Publications

Book Chapters

1. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2012. Enhancing Dependency Analysis by Combining Specific Dependency Parsers. Hybrid Intelligent Systems: Concepts and Applications (Book Chapter). iConcept Press, Ltd.

Publications

Conference Papers

1. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2010. A Feasibility Study on Low Level Techniques for Improving Parsing Accuracy for Spanish Using Maltparser. In Proceedings of **SETN 2010**. Athens, Greece. Springer LNAI 6040. 39-48.
2. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2010. Towards an N-Version Dependency Parser. In Proceedings of **TSD 2010**. Brno, Czech Republic. Springer LNAI 6231. 43-50.
3. **Miguel Ballesteros**, Jesús Herrera, Virginia Francisco and Pablo Gervás. 2010. Giving Shape to an N-Version Dependency Parser. In Proceedings of **IC3K-KDIR 2010**. Valencia, Spain. INSTICC. 336-341.
4. **Miguel Ballesteros**, Susana Bautista and Pablo Gervás. 2010. Text Simplification Using Dependency Parsing for Spanish. In Proceedings of **IC3K-KDIR 2010**. Valencia, Spain. INSTICC. 330-335.
5. Pablo Gervás and **Miguel Ballesteros** 2011. A Proposal for a Spanish Surface Realisation Shared Task. In Proceedings of **ENLG 2011**. Nancy France. ACL Onthology. 212-216.
6. **Miguel Ballesteros**, Virginia Francisco, Alberto Díaz, Jesús Herrera and Pablo Gervás. 2012. Inferring the Scope of Negation in Biomedical Documents. In proceedings of **CICLING 2012**, New Delhi, India. Springer LNCS 7181. 363-375.
7. **Miguel Ballesteros**, Alberto Díaz, Virginia Francisco, Pablo Gervás, Jorge Carrillo de Albornoz and Laura Plaza. 2012. UCM-2: a Rule-Based Approach to Infer the Scope of Negation via Dependency Parsing. *SEM Shared Task 2012. Resolving the Scope and Focus of Negation, (***SEM 2012**) Montreal, Canada. ACL Onthology. 288-293.
8. **Miguel Ballesteros** and Joakim Nivre. 2012. MaltOptimizer: An Optimization Tool for MaltParser. In Proceedings of the System Demonstration Session of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics (**EACL 2012**). Avignon, France. ACL Onthology. 58-62.
9. **Miguel Ballesteros** and Joakim Nivre. 2012. MaltOptimizer: A System for MaltParser Optimization. In Proceedings of the Eighth International Conference on Language Resources and Evaluation (**LREC 2012**), Istanbul, Turkey. 2757-2763.

Research Stays and Seminars

Research Stays:

1. Uppsala University, Sweden. Supervised by **Joakim Nivre**. 3rd October 2011 - 23rd December 2011.
2. Uppsala University, Sweden. Supervised by **Joakim Nivre**. 19th March 2012 - 19th April 2012.

Seminars and Invited Talks:

1. UCNLG+Eval - The 4th UCNLG Workshop: Language Generation and Evaluation, Edinburgh, United Kingdom. Invited by Anja Belz. 31st July 2011. Talk Title: *A Proposal for a Spanish Surface Realization Shared Task*.
2. Seminars of the Department of Linguistics and Philology, Uppsala University, Sweden. Invited by Joakim Nivre.
   14th October 2011.
   Talk Title: *Enhancing and Applying Dependency Analysis in Different Topics*.
3. Department of Computer Science, Stockholm University, Kista, Sweden. Invited by Hercules Dalianis. 20th October 2011. Talk Title: *Enhancing and Applying Dependency Analysis in Different Topics: Negation, Speculation and Text Simplification*.
4. Seminars of the Department of Linguistics and Philology, Uppsala University, Sweden. Invited by Joakim Nivre. 30th March 2012. Talk Title: *MaltOptimizer: A System for MaltParser Optimization*.

Conclusiones

- De la rama de Estudio del análisis de dependencias:
  - Concluímos que los corpora de entrenamiento pueden optimizarse de dos maneras.
  - Priorizar las frases largas.
  - No priorizar el tamaño final del corpus.
- De la rama de Mejora del análisis de dependencias.
  - Mostramos dos experimentos que se centran en la raíz de los árboles y subárboles de dependencias.
  - El segundo puede cambiar la manera de pensar de los investigadores sobre la raíz de los árboles, lo que lo convierte en muy importante.

Conclusiones

- De la rama de Optimización:
  - Hemos producido un sistema que es capaz de optimizar modelos de análisis.
  - Esos modelos pueden incluso producir resultados mejores que modelos optimizados manualmente por expertos en el área.
- De la rama de Aplicación:
  - Demostramos que las estructuras de dependencias son realmente útiles para resolver distintos problemas de PLN.

Thanks for your attention.