Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

DEVELOPING RELIABILITY METRICS AND VALIDATION TOOLS FOR DATASETS WITH DEEP LINGUISTIC INFORMATION

Sérgio Ricardo de Castro

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2011

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

DEVELOPING RELIABILITY METRICS AND VALIDATION TOOLS FOR DATASETS WITH DEEP LINGUISTIC INFORMATION

Sérgio Ricardo de Castro

DISSERTAÇÃO

Projecto orientado pelo Prof. Doutor António Manuel Horta Branco

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2011

Agradecimentos

Embora tenha sido interessante explorar uma área que não a minha área de especialização nesta tese de mestrado, esta nova área trouxe consigo a necessidade do domínio de termos e conceitos com os quais não tinha tido qualquer contacto até à data.

Por isso e por todo o outro apoio que me foi dado, gostaria de agradecer ao meu orientador, o Prof. António Manuel Horta Branco, e a todos os actuais e antigos membros do grupo NLX que me auxiliaram neste processo, sem os quais não teria sido possível.

Gostaria também de agradecer à minha família, aos meus pais e às minhas avós, os quais sempre acreditaram mais no sucesso deste projecto do que eu próprio.

Dedico este trabalho às minhas avós que já há muitos anos me chamam engenheiro.

Resumo

Grande parte das ferramentas de processamento de linguagem natural utilizadas hoje em dia, desde os anotadores morfossintácticos (POS taggers) até aos analisadores sintácticos (parsers), necessita de corpora anotados com a informação linguística necessária para efeitos de treino e avaliação.

A qualidade dos resultados obtidos por estas ferramentas está directamente ligada à qualidade dos corpora utilizados no seu treino ou avaliação. Como tal, é do mais alto interesse construir corpora anotados para treino ou avaliação com o maior nível de qualidade.

Tal como as técnicas e as ferramentas da área do processamento de linguagem natural se vão tornando mais sofisticadas e tecnicamente mais complexas, também a quantidade e profundidade da informação contida nos corpora anotados tem vindo a crescer. O estado da arte actual consiste em corpora anotados com informação gramatical profunda, isto é anotação que contém não só a função ou tipo de cada elemento mas também os tipos das relações entre os diferentes elementos, sejam estas directas ou de longa distância.

Esta quantidade crescente de informação contida na anotação dos corpora torna a tarefa da sua anotação crescentemente mais complexa, daí existir a necessidade de garantir que este processo resulta em corpora da melhor qualidade possível.

No seguimento desta crescente complexidade, as técnicas utilizadas para o processo de anotação também tem sofrido alterações. A quantidade de informação a ser introduzida no corpus é demasiado complexa para ser introduzida manualmente, portanto este processo é agora conduzido por uma gramática computacional, que produz todas as possíveis representações gramaticais para cada frase, e de seguida um ou mais anotadores humanos escolhem a representação gramatical que melhor se aplica a frase em questão.

Este processo garante uma uniformidade no formato da anotação, bem como consistência total nas etiquetas utilizadas, problemas recorrentes em corpus anotados manualmente.

O objectivo desta dissertação é o de identificar um método ou uma métrica que possibilite a avaliação da tarefa de anotação de corpora com informação gramatical profunda, bem como uma aplicação que permita a recolha dos dados necessários

referentes à tarefa de anotação, e que calcule a métrica ou métricas necessárias para validação e avaliação da tarefa.

Com este objectivo em mente, foi inicialmente explorado o trabalho de fundo da tarefa de anotação, tanto na vertente linguística como na vertente de processamento de linguagem natural.

Na vertente linguística, devem ser realçadas algumas noções base, tais como a de corpus, que se trata de um acervo de material linguístico originário de múltiplas fontes, tais como emissões de rádio, imprensa escrita e até conversas do dia-a-dia.

Um corpus anotado é um corpus em que o material foi explicitamente enriquecido com informação linguística que é implícita para um falante nativo da língua, com o objectivo de auxiliar ao processamento do material por parte de máquinas.

A anotação de corpus por parte do grupo NLX está a ser feita recorrendo a um esquema de anotação duplamente cego, em que dois anotadores escolhem de um conjunto de possíveis representações gramaticais atribuídas a cada frase pela gramática LXGram, a que para si é a mais correcta. Estas representações são posteriormente adjudicadas por um terceiro anotador. O resultado desta adjudicação é a representação que integra o corpus anotado.

O foco deste trabalho é o de avaliar a qualidade e fiabilidade do material resultante deste processo de anotação.

O processo de anotação pode ser visto como o processo de atribuição de categorias a itens, neste caso, a atribuição de categorias ou informação linguística a palavras ou multi-palavras de uma frase. Neste caso concreto, dada uma lista de discriminantes semânticos, os anotadores devem decidir quais pertencem ou não à melhor representação gramatical de uma dada frase.

Na literatura, existem várias abordagens para a avaliação de anotação com esquemas de anotação simples, por exemplo, com anotação morfossintáctica (POS tagging), como é o caso do Cohen's Kappa (Cohen, 1960), ou k, e suas variantes, tais como o S (Bennett et al., 1954), π (Scott, 1955) ou o próprio k.

Todas estas métricas se baseiam na mesma ideia de que a taxa de concordância entre anotadores (*inter-annotator agreement*) pode ser calculada tendo em conta dois valores: a concordância observada (A_e), isto é a quantidade de informação em relação à qual os anotadores concordam; e a concordância esperada (A_o), ou seja a quantidade de informação que se esperaria obter entre os anotadores se a anotação fosse feita aleatoriamente.

Todas as métricas derivadas directamente do Cohen's Kappa, calculam também a taxa de concordância da mesma forma, recorrendo à fórmula: concordância $= \frac{A_o - A_e}{1 - A_e}$.

O ponto de divergência entre as diferentes abordagens está na maneira de calcular a taxa de concordância esperada. Estas divergências consistem na representação da taxa de concordância esperada através de diferentes distribuições estatísticas.

Existe outro tipo de métricas, normalmente utilizado para a avaliação de análises sintácticas que também são aplicadas neste tipo de tarefa. Métricas como são o caso do Parseval (Black et al., 1991) e do Leaf Ancestor (Sampson and Babarczy, 2003) que frase a frase comparam a análise sintáctica dada pelo analisador sintáctico automático com um padrão dourado (análise sintáctica considerada correcta para a frase).

Contudo, a complexidade da tarefa a ser avaliada exige não só uma métrica sólida, mas também que a sua granularidade seja suficiente para distinguir pequenas divergências que podem sustentar resultados que aparentam ser contraditórios.

Tendo em conta a tarefa a ser avaliada, a abordagem mais granular possível é a consiste em comparar individualmente cada decisão sobre cada discriminante para uma dada frase.

Portanto, visto que o objectivo é obter a maior granularidade possível, para a métrica desenvolvida Y-Option Kappa, a taxa de acordo observado pode ser calculada pela razão entre o número de discriminantes com decisões idênticas, ou opções, e o número total de discriminantes disponíveis para uma dada frase.

Como cada discriminantes tem dois valores possíveis, isto é, ou pertence ou não à melhor representação gramatical, a taxa de concordância esperada pode ser considerada uma distribuição uniforme de decisões binárias, o que significa que o acordo esperado para caso de decisão aleatória será 0,5.

A métrica Y-Option Kappa é calculada através da mesma fórmula utilizada pelo Cohen's K e suas variantes.

A tarefa de anotação é auxiliada por um pacote de ferramentas linguísticas designado LOGON, pacote este que permite a anotação dinâmica de corpus, isto é as frases são analisadas dinamicamente pela gramática computacional conforme as decisões sobre os discriminantes são tomadas pelos anotadores. Isto permite ter acesso às representações gramaticais resultantes, possibilitando assim uma melhor percepção do resultado das decisões tomadas.

A informação resultante do processo de anotação é guardada em ficheiros de log que podem ser utilizados para reconstruir a representação gramatical resultante para a frase. Este pacote é bastante útil e fornece uma ajuda preciosa no processo de anotação. Contudo, os ficheiros de log guardam apenas a informação necessária para a reconstrução da representação gramatical final, o que resulta numa lista de discriminantes que pode ser incompleta para os propósitos de avaliação do processo de anotação.

Por exemplo, quando um anotador rejeita uma frase, ou seja, considera que não existe no conjunto possível de representações gramaticais uma que seja considerada correcta, apenas os discriminantes considerados até ao momento da rejeição são registados no ficheiro de log.

Para resolver este problema, algumas adaptações tiveram de ser feitas à ideia original da métrica Y-Options K para que esta fosse aplicável aos dados recolhidos.

Existem três casos gerais que resultam em conjuntos de informação concretos nos ficheiros de log. Estes três casos são:

- Cada anotador aceita uma representação gramatical como óptima para a frase: Todas as opções estão presentes e podem ser comparadas correctamente
- Pelo menos um dos anotadores rejeita qualquer representação gramatical para a frase: Existe apenas uma lista parcial das opções tomadas (para esse anotador).

Para resolver estes casos, são estimados sobre os casos em que toda a informação está disponível valores médios que são depois aplicados a casos em que a informação não esteja disponível. A métrica é assim calculada frase a frase, e o resultado final apresentado é a média aritmética da métrica para todas as frases.

Foi desenvolvida uma aplicação que permite através dos ficheiros de log determinar o valor da métrica, bem como alguma informação adicional para auxílio da tarefa de adjudicação.

Um objectivo futuro seria o de alterar as aplicações do pacote LOGON, mais concretamente o [incr tsdb()] de modo a que este guarde todos os discriminantes para cada frase, podendo assim dispensar o cálculo de estimativas.

Palavras-chave: Processamento de linguagem natural, taxa de concordância entre anotadores, anotação de copora com informação gramatical profunda.

Abstract

The purpose of this dissertation is to propose a reliability metric and respective validation tools for corpora annotated with deep linguistic information.

The annotation of corpus with deep linguistic information is a complex task, and therefore is aided by a computational grammar. This grammar generates all the possible grammatical representations for sentences. The human annotators select the most correct analysis for each sentence, or reject it if no suitable representation is achieved. This task is repeated by two human annotators under a double-blind annotation scheme and the resulting annotations are adjudicated by a third annotator.

This process should result in reliable datasets since the main purpose of this dataset is to be the training and validation data for other natural language processing tools. Therefore it is necessary to have a metric that assures such reliability and quality.

In most cases, the metrics uses for shallow annotation or parser evaluation have been used for this same task. However the increased complexity demands a better granularity in order to properly measure the reliability of the dataset.

With that in mind, I suggest the usage of a metric based on the Cohen's Kappa metric that instead of considering the assignment of tags to parts of the sentence, considers the decision at the level of the semantic discriminants, the most granular unit available for this task. By comparing each annotator's options it is possible to evaluate with a high degree of granularity how close their analysis were for any given sentence.

An application was developed that allowed the application of this model to the data resulting from the annotation process which was aided by the LOGON framework.

The output of this application not only has the metric for the annotated dataset, but some information related with divergent decision with the intent of aiding the adjudication process.

Keywords: Natural language processing, inter-annotator agreement, corpora annotation with deep linguistic information.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document describes the work done to fulfill the requirements of the Projecto de Engenharia Informática (PEI), from the MA on Engenharia Informática, at the Department of Informatics of the Faculty of Sciences of the University of Lisbon (FCUL). This introductory chapter presents the motivation underlying the work carried out, its objectives and its contributions. The project timeline and the structure of the document are also presented.

All the work supporting the project reported in this document was conducted in the NLX—Natural Language and Speech Group of the Department of Informatics of the University of Lisbon, under the project SemanticsShare.

1.1 Motivation

Human language processing tools, such as part-of-speech (POS) taggers or parsers, depend on the availability of properly annotated corpora for training and evaluation. The trend in the development of the used tools is to use corpora bearing increasingly sophisticated linguistic information. Therefore, the task of annotating corpora with the necessary deep linguistic information has become increasingly complex.

Initially, the linguistic materials were annotated manually by human annotators, a scenario that is unpractical for complex annotation schemes and very large corpora. With the increase in size and complexity of annotated corpora, automated annotation backed up by manual correction was employed. This approach still required the human annotator to go through all markables correcting annotation errors and potentially introducing new ones.

To mitigate this problem and part of the complexity associated with this task, the annotation can be done with the help of a deep linguistic grammar that returns all possible parses for each sentence, which is then disambiguated by human annotators. The disambiguation process consists on one or more human annotators picking the correct parse for a sentence from its parse forest. As there is a great

deal of complex decision in this kind of task, to increase the reliability of the resulting annotated corpora, the task is performed by more than one annotator working independently from each other (typically two, in a double-blind scheme), and the eventual disagreements are adjudicated by yet a third expert.

The level of agreement between the annotators is usually referred to as inter-annotator agreement (ITA). So far, most of the work on this subject has assumed that the annotation was relatively shallow, not going beyond the Penn treebank (Marcus et al., 1993) representation of syntactic constituency and POS tags. Exact match of trees or POS-tags has been used where the annotations produced by each annotator many times are compared in a binary fashion: either they are exactly the same or completely different.

With a deep linguistic annotation scheme like the one used in the Redwoods HPSG treebank (Oepen et al., 2002a), a finer granularity of comparison is required, since between a complete match and total failure there is a whole range of possibility in between the different possible annotations.

The current exact match metric (Cohen, 1960) to compare the choices of all the annotators previous to adjudication has been improved by introducing weight to compensate random decision (Cohen, 1960) and annotator bias (Carletta, 1996). However, the granularity offered by this adaptations is still not enough when it comes to corpora annotated with deep linguistic information. Therefore, there is a need for a robust and more granular metric for validation and evaluation of annotation with this level of complexity, supported and made operational by the appropriate tools.

1.2 Objectives

The main objective of this dissertation is to devise an ITA metric based on enough granularity to allow the evaluation and validation of the process of manual disambiguation of corpora automatically annotated by a computational grammar with deep linguistic information.

The foreseen solution for this problem consists in developing a metric that takes into account not only the final decision the human annotators make (either accept or reject) when disambiguating between the grammatical representations of a sentence advanced by the deep computational grammar, but also and primarily the set of options that have to be made to reach that final decision.

This kind of granularity will make possible the distinction even between two very similar grammatical representations.

It is also an objective of this dissertation to present a tool that allows to compute this inter-annotator agreement metrics as well as additional information to help the

adjudication process.

1.3 Contributions

This documents presents the ITA metric Y-option Kappa, a metric that allows the evaluation of the level of agreement of annotators for annotated corpora, and hence its reliability, which has been automatically marked and subsequently validated manually by human annotators using a double-blind annotation scheme.

Unlike the metrics commonly used until now for the task, this metric does not simply look to the trees that are accepted or rejected by the annotators, but looks instead for the semantic discriminants that each annotators accepts or rejects during the task of disambiguation. These semantic discriminants can be seen as traits of each element of a given sentence, and by accepting one of them, the annotators trim the parse forest, eliminating all the trees that do not comply with it.

These decision on the 'yes' or 'no' values for semantic discriminants can be called options, and if two annotators take the same options for a given sentence, the final result will with no doubt be the same. Two such grammatical representations (a.k.a. annotations) for the same sentence can diverge only by a limited number of the total of options necessary for its construction. Hence, we can compare the set of options of each annotator and, by doing so, measure how alike were their decisions.

In the task of annotation, a tool that facilitates the disambiguation and adjudication tasks was used. This tool, LOGON treebanking environment, was designed for overall aiding the disambiguation task and the reconstitution of the resulting parse trees.

The output of this tool lacks some of the features that would be desired. For example, only the options directly picked by the annotator are recorded in the correspondent logs. This and other omissions make the gathering of all the necessary data difficult. In some cases, relevant data has to be estimated. Some solutions for this problem are provided.

Besides the ITA metric, a tool was developed that collects all the information made available by the annotation tool LOGON output, and compute the necessary accumulated data and the ITA metric, also producing a spreadsheet report with the result data to aid the adjudication task. This tool was tested with the resulting corpus of the version 2 and 3 of the SemanticShare project.

1.4 Document Structure

The remaining chapters of this document are structured as following:

- (Chapter 2) Background - Introduces the necessary information to understand the results obtained and the work described in this document.
- (Chapter 3) Related Work - Gives an overview of what has been done about the problem being addressed, reporting on the state-of-the-art of the ITA metrics.
- (Chapter 4) Agreement Coefficient for Deep Linguistic Parsing - Presents a new granular and robust ITA metric idealised to be used with deep linguistic parsing.
- (Chapter 5) Experimental Assessment - Documents the practical application of the metric described in Chapter 4 and the results obtained with this new metric.
- (Chapter 6) Implementation - Describes the development and implementation of a tool that allows the interpretation of the deep linguistic parsing disambiguation toolkit output to be gathered and used to calculate the practical implementation of the metric documented on Chapter 4 and 5.
- (Chapter 7) Conclusions and Future Work - Discusses the results reported in Chapter 5 in this dissertation and the possible work that can result from them.

1.5 Work Timeline

The work to be developed during the PEI was divided as follows:

- Introduction to the problem, state-of-the-art, investigation and understanding. - 1 Month.
- Preliminary investigation of plausible solutions or adaptations of existing metrics - 3 Months.
- Design and development of the necessary tools to gather the necessary information and compute the metrics - 2 Months.
- Writing of the dissertation - 3 Months.

Chapter 2

Background

2.1 NLP and Datasets

2.1.1 Natural Language Processing[1]

Natural Language Processing (NLP) is a multidisciplinary field which has as main concern the processing of human language by computers. The first steps in this area were given during the second world war, with the main focus in machine translation.

The two principal motivations behind NLP are:

1. The pursuit of advances in the linguistic theory. Which requires the implementation of formal systems which allow to ensure internal consistency and to reveal its formal complexity. This is usually referred to as the theoretical motivation.

   Examples of this motivational focus are syntactic formalisms such as Phrase Structure Grammar, lexical Functional Grammar and Head Driven Phrase Structure, among many.

2. A more practical motivation interested in the creation of technology based on scientific principles as base for application for tasks such as translation, information extraction or summarization.

   However, most of these tasks can not be solved by linguistic methods alone.

In most of these tasks language is not only involved as a formalism but also as the input encoding from what is usually called 'the world'.

Natural language is ambiguous by nature. This combined with the ability people have to recover missing information or resolve ambiguity when taking context into consideration make it robust and allow it to be used casually.

[1]The content of this section is strongly based on the introduction of The Oxford Handbook of Computational Linguistics (Mitkov, 2003).

However solving such ambiguity and recovering context goes beyond pure linguistic means, which introduces the necessity for the use of artificial intelligence to build models which can restore or recover such information.

2.1.2 Corpora

The ability to identify and replicate linguistic information requires the usage of models. These models require training and for evaluation to ensure consistency and quality of both the themselves and their output, operations these which require language datasets as input.

A corpus is such a dataset. It is generally a large body of linguistic material, typically composed of attested language utterances. These utterances can originate from a multitude of material such as radio news broadcasts, published writing and every day conversations.

It is required for the corpus to be machine-readable. For example paper of voice based data has to be converted into a format the machine can understand.

The term corpus should not however be applied to any collection of text. A corpus is a well-organized collection of data with well defined boundaries of a specific *sampling rate*. it should also be representative of the domain or linguistic features to be studied.

A sampling frame is a guideline for the construction of a corpus. Corpus can be **monolingual corpora**, which means the corpora is constituted by material from only one language. Or a **comparable corpora** which means there are a series of monolingual corpora from different languages constructed using the same sampling frame, and similar representativeness, allowing for contrast study between different languages. Other way to compare languages is by constructing a **Parallel corpora**, in this case, the corpus is collected in one language and afterwards translated into the remaining languages to be studied. This allows for sentences to be compared side by side.

Corpora are fundamental both for NLP task execution and development of new technologies. However most tasks require more than a simple corpus, they require a corpus with additional information or analysis.

2.2 Treebanks

2.2.1 Annotated Corpora

An annotated corpus usually starts as a raw corpus[2] which is enriched in order to aid machine usage and understanding. The annotation process does not introduce

[2]The corpus without annotation is usually referred to as raw corpus.

new information from the human who is a native speaker[3] point of view, it only make the implicit information explicit. This encoded information allow for humans who lack the necessary meta-linguistic skills as well as computers to have access to such information without having the ability to make such linguistic analysis.

An annotated corpus has other advantages over a raw corpus from which the main ones are:

Ease of exploitation The information encoded into annotated corpora can turn fairly complex tasks into simple tasks. For example, retrieving all the nouns from every sentence in the corpus require a certain degree of linguistic knowledge about a given language or parsing capabilities of a computer, however if the corpus is correctly annotated, the tasks solution is trivial.

Reusability and Multi-functionality Once a corpus has the linguistic analysis encoded into it, this data can be used repeatedly and for different purposes without requiring the time consuming analysis to be repeated.

Explicit analysis A specific analysis is imposed into the corpus by the mean of the encoded information added to it by the responsible annotator.

Corpus annotation can be achieved either entirely automatically, by a semi-automated process, or entirely manually. Tools such as part-of-speech taggers or lemmatizers are so reliable that can be considered a fully automated approach to annotation. The wholly automated annotation process does inevitably mean a series of errors in the corpus, however the error rates associated with taggers such as the LX-Tagger(Branco and Silva, 2004) is of 3%, which is the state-of-the-art this kind of analysis.

Most NLP tools are not sufficiently accurate to allow fully automated annotation. However, they can be sufficiently accurate as to make correcting their automatic annotation faster than entirely annotating the corpus by hand. This was how the Penn Treebank (Marcus et al., 1993) was constructed, where the part-of-speech information of the corpus was first annotated by a computer and then corrected by human analysts. Fully manual annotation is a very slow tasks, which makes it only useful when no NLP applications are available or their accuracy is too low for it to be viable to manually validate its output.

2.2.2 The treebanks

A treebank is a type of annotated corpus, where the sentences are stored in syntactic tree format. This type of annotated corpus is widely used for NLP corpora, mostly

[3]has the necessary linguistic knowledge to make such a analysis.

because it information that make it easy for humans to read without the aid of special tools.

The most well known and broadly used treebank for English is the Penn Treebank (Marcus et al., 1993) which is a corpus mainly made of Wall Street Journal text. The Penn Treebank annotation was divided into two parts, the first was a automatic part-of-speech assignment; the second part consists in the manual correction of the part-of-speech tagging by annotators. This manual correction part of the annotation task has been proven to introduce less errors than a fully manual annotation scheme.

A treebank is a useful and important tool for the developing and utilization of NLP tools, however in some cases the constituency information present in such corpus is deemed to be insufficient. For example, the constituency annotation has no reference of long distance relations between words within a sentence. For more sophisticated schemes of annotation there is the necessity for the process of annotation to be aided by a grammar able to annotate the corpus with deep linguistic information.

2.3 Deep linguistic parsing

Deep linguistic parsing or annotation such as the Redwoods Treebank Oepen et al. (2002a) or the LXDeepBank Branco et al. (2010) differ from shallower treebanks in that they annotate the corpus with as much linguistic information as it is viable to extract from the text. For instance, this process can yield richer structural representations which can capture long-distance dependencies, syntactic and semantic features, or underlying predicate-argument structure directly, and this characteristics are fundamental for a more sophisticated analysis and processing. This kind of analysis result in a highly complex but complete representation of linguistic information. An example is illustrated in Image 2.1 where the deep linguistic analysis of the sentence "Todos os computadores têm um disco" ("Every computer has a disk") is shown in a Attribute-value matrix (AVM)format which we will describe further in the next section of this document.

The annotated corpora resulting from this approach are generally identified not only by their content but also by the computational grammar used in their analysis, this grammars are generally based on one of the many different theoretical models, mostly diverging in their linguistic information representation theory.

Such approaches as Combinatory Categorical Grammar (CCG) (Steedman and Baldridge, 2005), Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), Lexical Functional Grammar (LFG) (Falk, 2001), or Tree-Adjoining Grammar (TAG) (Joshi et al., 1975), are well known grammatical frameworks. In this project we use the HPSG framework, so this is the framework we will take into

consideration.

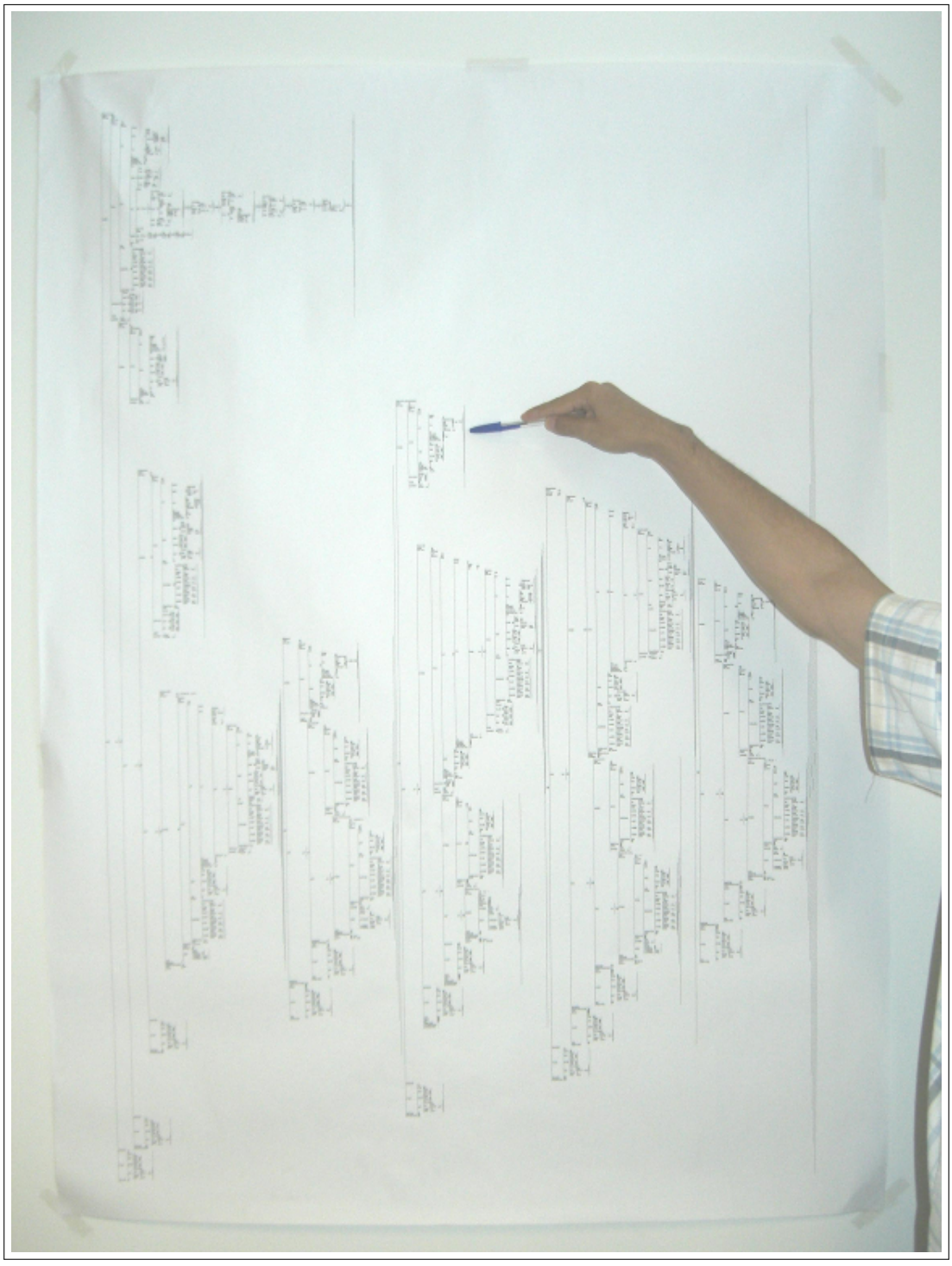


Figure 2.1: The HPSG in AVM format representation of the sentence ”Todos os computadores têm um disco” (”Every computer has a disk”), in font size 6. The arm and pen are included to give some perspective of the size of the structure.

2.4 Head-Driven Phrase Structure Grammar[4]

The HPSG framework follows the notion that a finite characterization of a language, represents the potential knowledge a subject has about possibly unlimited linguistic entities. That knowledge can be associated either to linguistic entities of the language for which the subject is a competent user or to entities external to that language.

Since language is a cognitive faculty, it follows the functional assumptions of cognitive science, under the assumption that cognitive processes can be compared to information processing which is better modeled by Church-Turing thesis result. With this in mind, a grammar for any language L is compatible with a parsing algorithm that, for a linguistic entity, allows the decision if it belongs to language L. Grammars are specially suited for the modulation of linguistic behavior and processing of linguistic information, hence a grammar is compatible with a parsing algorithm, even if not efficient in terms of computational complexity, it is at least tractable under typical conditions of natural mental parsing.

Grammar also allow the usage of models for partial processing of linguistic entities and flexible articulations between the sub-modules such as phonology, morphology, syntax and semantics. This allows the integration of different sub-models in a incremental way, resulting in a more complete grammar. The HPSG grammars in lato sensu can be divided into three symbolic structures, the linguistic principles, the grammar rules and the lexicon entries which are normally not considered to belong to a grammar in stricto sensu.

Since the HPSG model is lexical based, the lexicon entries have far more information than simple list entries. The lexical entries are richly structured, where individual entries are marked with types. This types are organized into a type hierarchy.

Type hierarchies are partial orders and Type Feature Structures (TFS). A type hierarchy is an acyclic graph in which the nodes are labeled with type labels and where a node n_2 which has an arc that starts at node n_1 is a subtype of n_1, and therefore more specific than the n_1. Figure 2.2 shows the type hierarchy for a small toy grammar built by Copestake (2002). All lexical types of this grammar are subtypes of the type *top*.

The type constraints define what are the characteristics of each type in the hierarchy, that is, it consists on associating to each type the constraints which define the most appropriated type structure for that type entity. The type hierarchy is a taxonomic tree where each type inherits the type constraints of its super-types. The type constraints fulfill the following properties:

[4]This section was strongly based on Chapter HPSG: Arquitectura(Branco and Costa, pear) from the Abordagens Computacionais da Teoria da Gramática book.

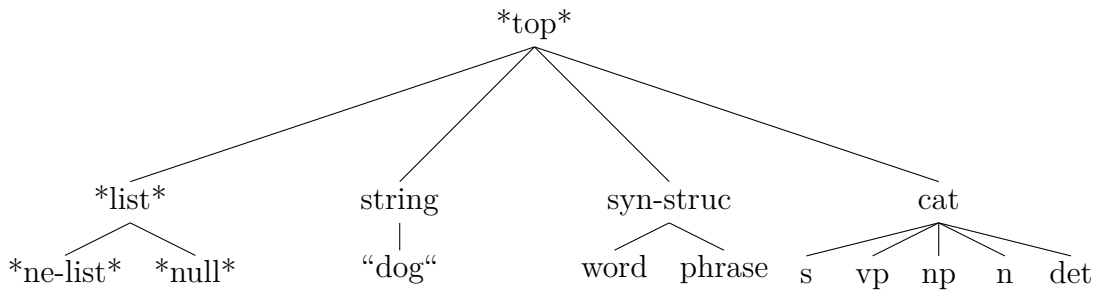


Figure 2.2: Type Hierarchy example.

Consistent inheritance All the constraints from any type parent types are inherited, hence any local constraints[5] must be compatible with the inherited informations, and in the case of multiple inheritance, the parents must unify.

Maximal introduction of features Any feature must be introduced at a single point in the hierarchy. That is, if a feature, F, is an appropriated feature for some type, **t**, the feature cannot be appropriated for a type which is not a descendant of **t**, and should not be introduced anywhere else in the type hierarchy.

Well-formedness of constraints All full constraint feature structures must obey the following properties:

Constraint Each substructure of a well-formed TFS must be included by the constrains of its root structure.

Appropriated features The top-level features[6] for each substructure of a well-formed TFS must be appropriated to the type of the root node substructure.

This constraints are expressed in a specific notation called attribute-value matrices (AVM). AVMs are matrices with two columns, in one we have the attributes and in the second their values. The values of the attributes themselves can be AVMs.

$$\begin{bmatrix} type & \\ FEATURE_1 & avm_1 \\ FEATURE_2 & avm_2 \\ \dots & \dots \\ FEATURE_n & avm_n \end{bmatrix}$$

Figure 2.3: Example of an attribute-value matrix (AVM)

[5]Local constraints are feature structures directly specified in the descriptions.
[6]Top-level features are the features that label arcs starting from the root node of a structure.

There are constraints relative to the Universal Grammar and constraints specific for the grammar language or its language family, and there are also the lexical rules which include constraints with distinct formal properties, used to capture lexical entry generalizations. This rules are ”meta-descriptions” since they refer to entity specification and not the phrase structure.

2.4.1 Grammar computational machinery

The linguistic knowledge represented by a grammar allow the fulfillment of two computational tasks:

Generation: given a semantic representation, obtain set of sentences for which the grammatical representation includes that semantic representation, that is, a list of sentences expressing the meaning encoded in the original semantic representation.

Parsing: given an expression, verify if that expression belongs to the grammar, if so, obtain a set of grammatical representations that fit such expression.

The computational implementation of this formalism relies greatly on two mechanisms, the **unification** and **parsing**. The parsing mechanism allows the decision on whether an expression belongs to the set of expressions intensionally defined by the grammar as forming a language or not. The unification mechanism allows for the construction of representations by the combination of a partial compatible representation and a possible instantiation of variables in that representation.

There is a third task that can be computationally performed by the grammar, after the parsing task is complete, and that is the possibility to resolve ambiguity. The task of **ambiguity resolution** consists on identifying from a set of possible grammatical representations, the one that most likely is conveyed by the initial expression. To make this possible, an automatic classification system base on stochastic parameters are used. This procedures require corpora made up of sentences annotated with the most likely grammatical representation for their context.

2.4.2 Minimal Recursion Semantics

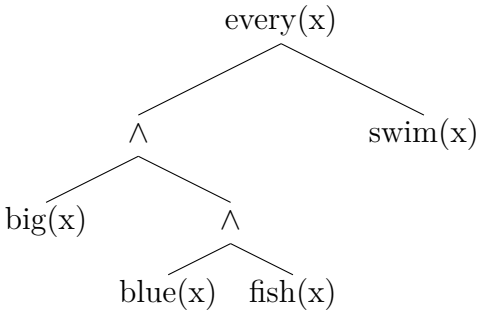
The semantic information of the HPSG framework grammars is structured using the Minimal Recursion Semantics.

Minimal Recursion Semantics (MRS)(Copestake et al., 2005) is a formalism that allows the representation of under-specified semantics. The MRS is not a theory of semantics but rather a format or meta-level language for semantic structures

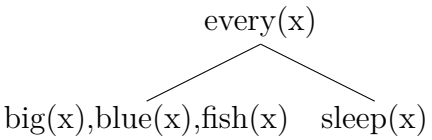
representation. A MRS representation of a given sentence allows the derivation multiple representations which are the possible interpretations of a sentence.

The basic structural unit of an MRS are the elementary predications (EP). Elementary predications are semantic relations and their required arguments, e.g. like(x,y). Hence phrases can be represented in a logic formula as for example the sentence *little blue ball* which can be represented as: $\lambda x(little(x) \wedge (blue(x) \wedge ball(x)))$ or $\lambda x(little(x) \wedge (blue(x)) \wedge ball(x))$ in logical form, and contains the three elementary predications $little(x)$, $blue(x)$ and $ball(x)$ connected by logical operators $\vee$ (or) and $\wedge$ (and). Since the logical operation *and* is associative, the differences between the two representations for the sentence *little blue ball* can be neglected in terms of conditional semantics. Knowing this, both representations can be represented just by $little(x), blue(x), ball(x)$.

This is equivalent to converting a tree structure to a tree of depth 1.For example the tree

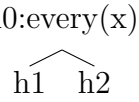


Can be merged into the following minimally recursive structure



The links of the tree structure underlying each non-minimally recursive representation are replaced by tags (or handles) which identify an EP and a potential scopal argument position.

A tree with the handles as nodes and a list of elements associated to each handle are created. For the previous example it would be:



$$h1 = (big(x), blue(x), fish(x)) h2 = (swims(x))$$

This results in a flexible enough structure to accommodate scope while still retaining a minimal structure.

2.4.3 Types and feature structures

In the HPSG framework, linguistic entities are organized into types. Each linguistic object has a specific type. All syntactic constituents are an instance of the type *sign*.

Each type has a set of features associated to it. This features are pairs of attributes and respective values. The type *sign* for example has the attributes PHON (phonology) and SYNSEN (syntax-semantics).

The values for this attributes are also linguistic objects, and therefore have types associated to them. The attribute PHON values contain information related with the spelling of the respective constituent. The SYNSEM value contains semantic and category information and has *synsem* as type. The *synsem* type has in turn two attributes: LOCAL and NON-LOCAL. The NON-LOCAL attribute has information related with the long distance dependencies and the LOCAL attribute old information related with semantic and syntax in the attributes CONT (content) and CAT (category) respectively.

The types *sign*, *synsem*, *loc* can be represented by the AVMs in figure 2.4.

$$\begin{bmatrix} sign & \\ PHON & list(str) \\ SYNSEM & synsem \end{bmatrix} \begin{bmatrix} synsem & \\ LOCAL & loc \\ NON\text{-}LOCAL & non\text{-}loc \end{bmatrix} \begin{bmatrix} loc & \\ CAT & cat \\ CONT & cont \end{bmatrix}$$

Figure 2.4: AVM representations for the *sign*, *synsem*, *loc*, *cat* and *val* types.

The grammatical object *sign* is represented by AVM in Figure 2.5.

The value of the attribute of PHON of the presented AVM is a list of items referent to information such as phonology and orthography. The values of attributes SUBJ SPR and COMPS are also lists of objects, in this case of the type *synsem*.

2.4.4 Linguistic Knowledge Builder

The Linguistic Knowledge Builder (LKB) is an open-source development environment for constraint-based grammars. This environment provides a GUI, debugging tools and very efficient algorithms for parsing and generation with the grammars developed with its aid.

A considerable number of broad coverage grammars for diverse languages have been developed in the LKB, such as the English (Copestake and Flickinger, 2000), German (Crysmann, 2007) and Japanese (Siegel and Bender, 2002), and all obtained good results. Other important feature of the LKB is that the grammars developed in the LKB are supported by the PET parser (Callmeier, 2000), which allows a faster parsing time due to the compiling of the grammars into a binary format.

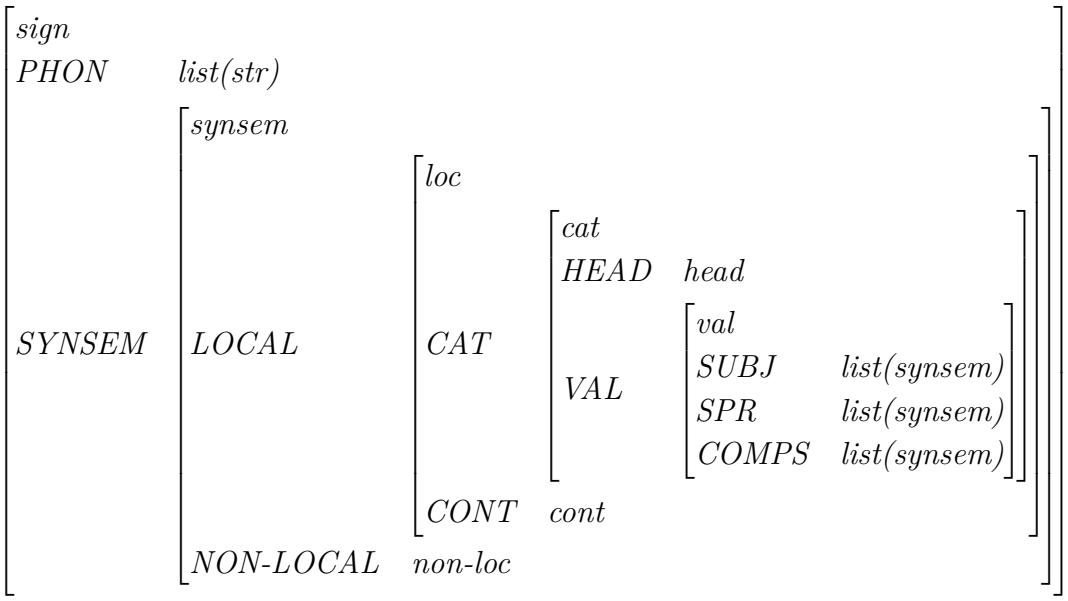


Figure 2.5: AVM representation of the grammatical object *sign*.

Note that there are more attributes under the features CONT and NON-LOCAL, which are appropriated to *cont* and *non-loc*, those are omitted here, to make the AVM easier to read.

2.4.5 LOGON

The LOGON infrastructure (Lønning et al., 2004) is a collection of software and other linguistic resources with the objective to facilitate experimentation. It was originally developed with machine translation in mind but it combines an impressive number of open-source tools that are useful in the construction of HPSG grammars and treebank annotation. Tools such as the LKB, the PET parser Dr et al. (2001) and the [incr tsdb()] Oepen (2001) software systems.

Even though this tools exist and can be used separately, the LOGON infrastructure offers the combination of all of them in one single place, which results in a good environment for dynamic annotation with deep linguistic grammars.

The LOGON infrastructure will be reviewed in more detail on Chapter 5.1.

2.4.6 Summary

The HPSG framework is theoretical rich, with formal rigor and computational versatility. It is a robust model based on two essential components: an explicit, highly structured representation of grammatical categories, encoded as typed feature structures, of which complex geometry is motivated by empirical considerations against the background of theoretical concerns such as locality; a set of descriptive constraints on the modeled categories expressing linguistic generalizations and declarative characterization of expressions admitted as part of the natural language. Be-

cause of this characteristics it is a widely used grammatical framework, both in linguistics and natural language processing, this resulting in the availability of both development tools and computational grammars for different languages.

2.5 LXGram

The LXGram (Costa and Branco, 2010) is the first general purpose grammar for deep linguistic processing of the Portuguese which contains a thorough and principled linguistic analysis of sentences, including their formal semantic representation. It was and still is being developed by the NLX group, and is the grammar used to generate the corpora used in this document.

2.5.1 Scope and Design Features

The LXGram grammar follows the grammatical framework HPSG and is developed using the LKB system. It is based on hand coded linguistic generalizations supplemented by a stochastic model for parsing ambiguity resolution.

The MRS format is used for representation of meaning, which supports scope underspecification. Semantic representations provide an additional level of abstraction by abstracting word order and language specific grammatical restrictions.

The LXGram supports a wide range of linguistic phenomena, such as coordination, subordination, long distance dependencies, modification and many subcategorization frames. LXGram contains to this moment, 64 lexical rules, 101 syntax rules, around 850 lexical leaf types (which determine syntactic and semantic properties of lexical entries), and 35000 lines of code (excluding the lexicon), also, its lexicon contains over 25000 entries. Supporting both European and Brazilian Portuguese, containing specific lexical entries to either of them, also converging both European and Brazilian syntax. The grammar is still in development, which opens room for constant improvement.

Because during the generation process the grammar can generate multiple solutions for a given sentence, a statistical disambiguation model was used to select the most likely analysis for that sentence. This model was trained with a dataset of 2000 sentences of newspaper text, using a maximum entropy algorithm.

2.5.2 Coverage

A good method to evaluate a grammar is to assert its coverage of spontaneous text. That is, given text generated without this special task in mind, for example journalistic text or reports, this text is fed to the grammar and the proportion of it to receive a grammatical representation is asserted.

With this objective in mind, an experiment was conducted with the objective of asserting the LXGram coverage. This experiment consisted on using the LXGram to parse a random selection of 30000 sentences from two publicly available newspaper corpora, CETEMPúblico and CETENFolha which contain text from "O Público" and "Folha de São Paulo" newspapers respectively, and over 60000 sentences extracted from Portuguese Wikipedia, previously tagged by the part-of-speech tagger and morphological analyzer (Silva, 2007), the LXGram obtained a coverage of 32% (32% for Wikipedia, 28% CETEMPúblico and 37% for the CETENFolha). The results of other computational HPSGs, as mentioned by Zhang et al. (2009) are of 80.4% of newspaper coverage for the English grammar, 42.7% for the Japanese grammar and 28.6% for the German grammar. However, all of this grammars have been in development for over 15 years now, and they LXGram has only been in development for 4 years. The Spanish Resource Grammar (Silva, 2007), a Spanish HPSG grammar, a language quite similar to Portuguese, and approximately as old as LXGram, has a reported coverage of 7.5%.

Other important method for grammar evaluation is the measuring of its accuracy. The accuracy metrics usually involve a golden standard, which is the considered correct grammatical representation for a given sentence. Since in this case there is no golden standard, a fluent user of the grammar language takes into account the sentences that receive a representation from the grammar and asserting the amount of this representations considered correct.

With the objective of asserting the accuracy of the LXGram a sample of the first 50 parsed sentences of the CETENFolha sub-corpus. From this sentences, 20 were correctly parsed, and the preferred reading was the one chosen by the disambiguation model. 10 sentences received the correct parse, but did not receive the preferred reading from the disambiguation model. For the remaining 20 sentences, 12 were affected by part-of-speech tagger or the morphological analyzer errors, and 8 were due to genuine limitations in the grammar or the disambiguation model (for instance, lack of some sub-categorization frames for some words in the lexicon). This data indicates that a good portion of the parsed sentences get a correct representation (60%) and are disambiguated correctly (40%).

2.5.3 Summary

This coverage experiment data shows that LXGram is robust grammar with a good coverage and is still in development, which opens room to improvement and growth. This factors make it a good grammar for the creation of a treebank of sentences parsed with it and manually disambiguated, which can be used as training corpora for many NLP tools for the Portuguese language.

2.6 CINTIL LXDeepBank

The usage of deep linguistic grammars in the process of construction of annotated treebanks is becoming essential in the area on NLP. The LXDeepBank or CINTIL LXDeepBank (Branco et al., 2010), is the corpus developed by the NLX group, where sentences are annotated with fully fledged linguistically informed grammatical representations that are produced by a deep linguistic processing grammar, thus consistently integrating morphology, syntactic and semantic information. At version 3 (the current version when this document was written) the LXDeepBank has 5422 fully annotated sentences.

Since LXDeepBank is the corpus generated with the LXGram and subsequent NLX tools, it is the corpus used for the testing and all the data examples of this document, and therefore is important to get to know it a little better.

2.6.1 Construction of the LXDeepBank

The LXDeepBank construction process consists on the usage of a computational grammar, in this case the LXGram, producing all the possible [7] grammatical analysis for the sentences, a parse forest of the corpora. Since a single sentence can have many grammatically valid interpretations, and therefore a very large parse forest, it is necessary to most correct interpretation for the given sentence, for this purpose, human annotators, select for each sentence the parse they consider the most correct interpretation for that sentence using a double blind annotation followed by adjudication scheme. In our case the annotation is done by two annotators and then adjudicated by yet another annotator.

Since going through all the grammatical analysis and picking one would be an extremely complex task and prone to error, we use a tool designed for this function,[8] LOGON infrastructure (Oepen et al., 2007). This tool presents, for each sentence of the corpora, a set of **semantic discriminants** relevant to the possible grammatical representations of the sentence, which can be seen as binary predicates. The annotators can consider this semantic discriminants to be true or false for the grammatical representation they find most correct for this sentence.

Semantic discriminants describe the semantic relation between constituents or a constituent and certain semantic properties, like gender, number, or tense.

The selection or not of a semantic discriminant results in the trimming of the available parse forest. For example, for the sentence in example (1) the annotator

[7] For performance reasons in practice only the 250 most likely grammatical interpretations are generated.

[8] The LOGON environment has other functions including machine translation but this is the one relevant for this subject.

could classify as true or false the semantic discriminant that describes 'binóculos' as modifier of 'vê' and by doing so, all the parse trees that represented 'binóculars' as being the modifier of any other constituent of the sentence would be removed from the parse forest.

(1) O João vê a Maria com os binóculos
The João sees the Maria with the biconulars
João sees Maria with the binoculars

The selection of semantic discriminants continue until there is only one tree left in the parse forest, or no more semantic discriminants to select from. The resulting grammatical representation is the one the annotator considers the most correct for the considered sentence. In case of divergence between the decision of the annotators, a third more experiment annotator select the grammatical representation he thinks to be the most correct, eliminating the indecision. The LXDeepBank consists on a corpora of this grammatical representation.

2.6.2 The vistas

The amount of grammatical information encoded in a deep linguistic databank like LXDeepBank can be overwhelming both by its quantity and complexity, as it is evident in Figure 2.1, also it may lead to the information being stored in a too theory-specific format. With this in mind, a series of tools that allow the extraction of different vistas out of the LXDeepBank were developed. While sharing the same base corpora, each vista corresponds to an individual annotated corpora. The vistas or annotated corpora available are the: CINTIL Treebank, CINTIL DependencyBank, CINTIL Propbank and a CINTIL LogicalFormBank.

Each vista follows the linguistic options that comply with the current best practice, and are encoded in facto standards for data formats (viz. Penn Treebank, CoNLL, etc.).

CINTIL Treebank

The CINTIL Treebank is the corpora extracted from the LXDeepBank where only the constituency information is represented. Since the information was available, it was also made possible by the extraction tool to obtain some options that can be included or excluded from this representation, forming a set of possible sub-corpora with different levels of annotation. Some of the available options are:

- The nodes can be stripped of all grammatical functions except for the constituency tags in order to create a treebank equivalent to the Penn Treebank

in annotation level.

- Null subjects and ellipsis can be represented as additional empty nodes or specific node labeling.
- Multi-word expressions[9] can be kept in a single node under a pre-terminal node or be expanded with the addition of empty nodes.
- Morphological information such as POS tag, lemma and inflection features can be removed or appended to the leaves of the resulting tree.

The CINTIL Treebank graphical representation in Figure 2.6 contains grammatical functions and semantic roles, it also contains a null subject as an empty node with a leaf called *NULL*. The multi-word expression (Estados Unidos) has been expanded into two nodes. When appropriated the long-distance syntactic information has been left by means of slashed labels, such as, 'PP-M-LOC'.

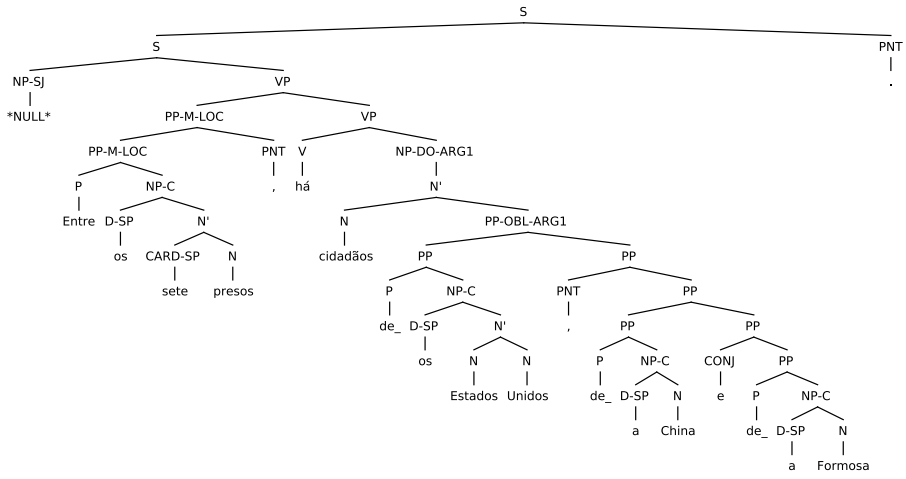


Figure 2.6: CINTIL TreeBank vista for the sentence "Entre os sete presos, há cidadãos dos Estados Unidos, da China e da Formosa" ("Between the seven prisoners, there are citizens of the United States, China and Taiwan")

CINTIL DependencyBank

The information represented in the CINTIL DepedencyBank is the grammatical relations between the constituents of the sentence. This information is stored in the CoNLL format, a tabular format where each entry corresponds to a word of the sentence, and each word includes fields such as lemma, head and POS. It is also

[9]Multi-word expression (MWE) is a lexeme made up of a sequence of two or more lexemes that has properties that are not predictable from the properties of the individual lexemes or their normal mode of combination.

possible to obtain a graphical representation of this structure, Figure 2.7 contains the CINTIL DependencyBank graphical representation for the same sentence as in the one in Figure 2.6 for the CINTIL TreeBank.

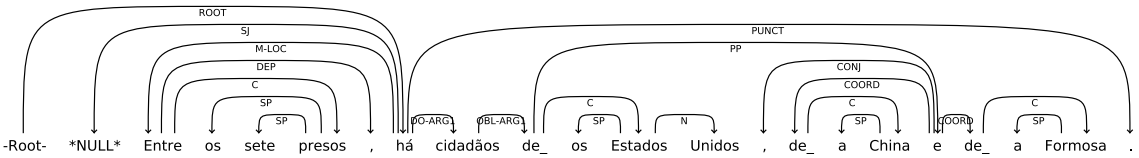


Figure 2.7: CINTIL DependencyBank vista for the sentence “Entre os sete presos, há cidadãos dos Estados Unidos, da China e da Formosa” (“Between the seven prisoners, there are citizens of the United States, China and Taiwan”)

CINTIL Propbank

Propbanks are treebanks whose the trees have their constituents labeled with semantic role tags. A propbank can be seen has a expansion of a treebank with additional grammatical information, that is, semantic categorization of phrases.

Since this information can be added to the treebank information, the representation is similar, only differing in the slashed labels, as it is the case of the label ’PP-M-LOC’ in Figure 2.6 where the LOC part is the semantic information.

CINTIL LogicalFormBank

The last vista, the CINTIL LogicalFormBank consists on the MRS representation of the corpora.

Since the semantic representation provided by the MRS is abstract to word order and language specific grammatical restrictions, it provides an additional level of abstraction. For example, the fact that the Portuguese verb *gostar* selects for a PP complement and its English counterpart *like* is a transitive verb is not visible in the semantics, since both cases correspond to an equivalent binary predicate.

2.6.3 Dynamic databank

Since the grammar used in the process of the construction of the LXDeepGram databank is in active development, and so the parsed portion of the corpora change over time, it is necessary to live with a dynamic databank as proposed by (Oepen et al., 2002b). Since the computational grammar keeps evolving and the covered part of the corpora keeps increasing.

This notion consists in building successive databanks where the unchanged data between computational grammar iterations is not recomputed, saving resources to deal with the new phenomenon covered by the computational grammar evolution.

2.6.4 Summary

The LXDeepBank is a sophisticated corpora for the Portuguese language with deep linguistic information, and its different vistas allow it to be used as training and evaluation corpora for a variety of different tools and applications. As the LXGram goes through development cycles and its coverage grows, so will the LXDeepBank size and phenomena.

Chapter 3

Related Work

Most natural language processing tools use statistical models to reach their objectives. They use models that require datasets for training and for performance evaluation. To ensure the availability of such datasets annotated corpora have to be developed. To have reliable, accurate and robust linguistic tools, reliable and accurate annotated corpora are required.

3.1 Data reliability

When using fully or partially manually annotated datasets, which are obtained through the process of associating linguistic items with their correct linguistic information, it is necessary to guarantee that the data resulting from this annotation is **reliable**. In this context, for data to be reliable, different annotators annotating the same items should **agree** by associating the same pieces of information to them (Krippendorff, 2004, Craggs and Wood, 2005). Different annotators consistently achieving similar annotation decisions indicate that the different annotators have achieved a similar understanding of the annotation task. And a better understanding of the annotation task leads to increased reliability of the annotated corpora.

As far as annotated corpora are concerned, the present trend is to use increasingly more complex annotation schemes that make the subsequent use of the corpora more productive, but make the annotation process much more complex.

To ensure the quality of the resulting dataset, advanced techniques have been used in the annotation process to try to reduce to a minimum the impact of possible annotator bias on the annotation. The most common is the double-blind technique.

3.1.1 Double-blind annotation

A **Blind experiment** is a scientific experiment where some or all of the participants in the experiment are prevented from knowing certain information that might lead

to conscious or unconscious bias on their part, thus invalidating results.

The **Double-blind** approach consists on applying the same technique as in the blind experiment, but it tries to eliminate bias from both the experimental subjects and the experimenters. That is, neither the experimental subjects nor the experimenters know who belongs to control groups or are actual experimentation subjects. This kind of experiment allows the information to be collected without a particular subject or observer bias.

In the NLP applications the double-blind experimentation technique concept is somewhat different. It is also used for tasks that can suffer from either personal or peer bias. However, there is no control group. It refers to two or more human annotators performing a task simultaneously, where the annotators have no knowledge of each other analysis, and cannot discuss it until the task is complete. In this document case study, it is used in the annotation of corpora and in automatic annotation disambiguation tasks. This adaptation of double-blind experimentation is called **double-blind annotation**.

Since it is unlikely that different annotators have the same bias or misappreciation of a given linguistic item, the effect of such personal bias or incorrections are mitigated by the comparison of the annotators results. This consists in the process of adjudication. The whole process is usually called **double-blind annotation followed by adjudication**.

The process of **adjudication** typically consists in a more experienced annotator reviewing the annotators analysis and in cases where there is disagreement, selecting the most correct analysis for such cases.

3.1.2 Agreement

In this setup, a close similarity between the analysis of multiple annotators demonstrates the **validity** of the annotation scheme, since valid annotation schemes are easier to understand, assimilate and apply by the annotators. The close similarity between the decisions of multiple annotators also attests the reliability of the resulting annotated dataset.

Therefore it is important to be able to quantify how similar the annotated dataset produced by different annotators are in order to evaluate the quality of the resulting annotated corpora.

3.2 Agreement Coefficients

Encouraged by the need to reliably quantify similarity between different annotations, several suggestions have been made, some of which have been adopted in the literature. Such quantification metrics are refereed to as **Agreement coefficients**,

since they try to measure the agreement between two or more sources of information, in this case annotators analysis. The two main approaches are introduced in the next two subsections.

Notation and concepts

To better understand the different concepts introduced in this section, it is necessary to begin by introducing the notation below and some of the most important concepts they are annotated to:

All the agreement coefficients assume that the datasets used consist of **items** $\{\mathbf{i} \mid \mathbf{i} \in \mathbf{I}\}$ (the units that can be annotated), a set of **categories** $\{\mathbf{k} \mid \mathbf{k} \in \mathbf{K}\}$ (what you can annotate the items with), and **coders** $\{\mathbf{c} \mid \mathbf{c} \in \mathbf{C}\}$ (or **annotators**) who assign to each item a unique category label.

Agreement

The notation A^X denotes **agreement** as calculated for the coefficient X, the agreement can be of two types. A_o refers to the **observed disagreement**, that is, the proportion of the equal attributions of item i to category k by both annotators over all the attributions. A_e is the **expected agreement**, the proportion of agreement if decisions were made by chance alone.

Disagreement

D_o represents the **observed disagreement** and is the proportion of items to which the annotators attributed different categories. While D_e is the **expected disagreement**, the proportion of disagreement if the annotators disagreed by chance alone.

Probabilities

$P(x)$ denotes the probability of x happening, whereas $\hat{P}(x)$ refers to the estimate of such probability to happen.

Quantities

The letter n denotes a exact number or quantity, the type depending entirely on the subscript. For example n_k refers to the number of times the category k was attributed and n_ik the number of times the item i was attributed to category k.

3.2.1 Cohen's Kappa and variants

The first approach presented consists in measuring the inter-annotator agreement by comparing how many times the annotators picked the same item or analysis, and in some cases take into account the agreement achieved by chance alone. This

approach is usually related to the Cohen's Kappa or κ (Cohen, 1960) and variants, and it is the most widespread in the literature.

Agreement Without Chance Correction

As Scott (1955) pointed out, the simplest way to measure the agreement between two annotators is by means of the percentage of agreement or **observed agreement** ($\mathbf{A_o}$). This is nothing more than the number of items to be marked on which the annotators agree divided by the total number of items to be marked. Accordingly, the agreement value arg_i for each item $i \in I$ is defined by: arg_i is 1 if two **annotators** assign the **item** i to the same **category k**, and 0 otherwise.

The observed agreement can be calculated, for all items $i \in I$, by computing:

$$A_o = \frac{1}{i} \sum_{i \in I} arg_i \tag{3.1}$$

A nice way to illustrate this metric is by using a very simple annotation scheme, for example, dialogue acts in information-seeking dialogues where an annotator can either choose **statement** or **info-request**, making this a binary decision. The results of DAMSL dialogue act scheme (Allen and Core, 1997) contain 100 utterances annotated by two annotators. Considering Table 3.1, the percentage agreement for this data set is obtained by summing up the cells on the diagonal and dividing by the total number of items so $A_o = (25 + 50)/100 = 0.75$ which indicates the annotators agreed on 75% of the assignments.

The result of this agreement metric is heavily dependent of the study it is used on, since some agreement is due to chance and the chance agreement itself it is affected by two factors that vary from one study to the next. First, as Scott (1955, page 322) pointed out, "[percentage agreement] is biased in favor of dimensions with a small number of categories." which means, the fewer categories an annotation scheme has, the higher the percentage agreement will be.

The second factor affecting percentage agreement across studies is the fact that it does not take into account the distribution of the item. This means a higher agreement if likely when one category is much more common than the other. This problem was raised by Hsu et al. (2003, page 207) and can be illustrated by the following example (Di Eugenio and Glass, 2004, example 3, page 98-99). In a particular domain, **statement** represents 95% of the utterances, and info-request only 5%. The expected agreement by chance would be $0.95 \times 0.95 = 0.9025$ that both annotators would both pick **statement** and of $0.05 \times 0.05 = 0.0025$ for **info-request**, so by chance the annotators would agree 90.5% of the utterances. Which would make, for instance, what would seem to be a high observed agreement of 90% to actually be worse than an agreement that would be expected by mere chance.

Chance-Corrected Agreement Coefficients

All the agreement coefficients presented below correct decisions by chance on the base of the same idea. This idea consists on calculating the agreement that would be expected by chance alone. For the sake of simplification, this is called **expected agreement** or $\mathbf{A_e}$. The agreement beyond chance alone can be obtained by calculating $1 - A_e$, and the agreement actually observed beyond chance can be calculated using $A_o - A_e$. The ratio between $A_o - A_e$ and $1 - A_e$, gives us the proportion of agreement beyond chance that is actually observed. This calculation is formalized by equation (3.2).

$$S, \pi, \kappa = \frac{A_o - A_e}{1 - A_e} \tag{3.2}$$

The most used and well-known agreement coefficients are S (Bennett et al., 1954), π (Scott, 1955), and κ (Cohen, 1960), and generalizations. All use the proportion (3.2) for agreement calculation, whereas Krippendorff's α is based on a related formula but expressed in terms of disagreement. All three coefficients consider similar values of agreement where $-A_e/1 - A_e$ corresponds to no observed agreement, 1 to perfect observed agreement and 0 corresponds to mere chance agreement (observed agreement = expected agreement). It is important noting that when observed agreement is below the maximum value ($A_o < 1$) the chance-corrected agreement will be lower than the observed agreement, because some agreement is always expected by chance.

All three coefficients consider the A_o as the proportion of items on which both annotators agree. What they differ in is on the model each uses for the chance agreement. Independence of the two annotators is assumed by all three, that is, given two annotators c_1 and c_2 the the chance of both agreeing on any given category k is the product of the chance of each of them picking that category: $P(c_1|k) \cdot P(c_2|k)$. Expected agreement is the sum of this product over all categories as captured by equation (3.3).

		Coder A		
		Stat	Ireq	Total
Coder B	Stat	25	15	40
	Ireq	10	50	60
	Total	35	65	100

Table 3.1: A simple example of agreement on dialogue act tagging.

$$A_e^S = A_e^\pi = A_e^\kappa = \sum_{k \in k} P(k|c_1) \cdot P(k|c_2) \tag{3.3}$$

The assumptions leading to the calculation of $P(k|c_i)$, the chance of a coder c_i assigning an arbitrary category to k is where the difference between the coefficients S, π and κ is.

The S coefficient considers that annotators decisions by chance alone correspond to a uniform distribution. This means that all categories are equally likely to be picked. That being so, for any two annotators c_m, c_n and any two categories k_j, k_l, $P(k_j|c_m) = P(k_l|c_n)$. If the annotators have to assign an item to one of $\mathbf{k}$ categories, the probability of assigning one by chance is $\frac{1}{\mathbf{k}}$. The expected agreement can therefore be calculated using equation (3.4).

$$A_e^S = \sum_{k \in K} \frac{1}{k} \cdot \frac{1}{k} = k \cdot (\frac{1}{k})^2 = \frac{1}{k} \tag{3.4}$$

For the coefficient π, annotators acting on chance alone would get similar distributions to those found on the actual world, and the best estimate of that is the proportion calculated by taking into account the number of assignments to category k by both annotators n_k, divided by two times the numbers of items to be annotated, so we have $\hat{P}(k) = n_k/2\mathfrak{i}$. Since with π it is assumed that annotators act independently, we can calculate the expected agreement as illustrated in equation (3.5).

$$A_e^\pi = \sum_{k \in K} \hat{P}(k) \cdot \hat{P}(k) = \sum_{k \in K} (\frac{n_k}{2\mathfrak{i}})^2 = \frac{1}{4\mathfrak{i}^2} \sum_{k \in K} n_k^2 \tag{3.5}$$

Finally, for the κ coefficient, each annotator would have its own individual distribution reflecting its own personal bias. To assert this distribution, the annotator's prior distribution is considered, that is, the proportion of items annotator c_i assigned to category k or $n_{c_i k}$ divided by the number of items, that is, $\hat{P}(k|c_i) = n_{c_i k}/\mathfrak{i}$. The expected agreement is therefore the sum of the joint $\hat{P}$'s for all categories k, as illustrated on equation (3.6).

$$A_e^\pi = \sum_{k \in K} \hat{P}(k|c_1) \cdot \hat{P}(k|c_2) = \sum_{k \in K} \frac{n_{c_1 k}}{\mathfrak{i}} \cdot \frac{n_{c_2 k}}{\mathfrak{i}} = \frac{1}{\mathfrak{i}^2} \sum_{k \in K} n_{c_1 k} n_{c_2 k} \tag{3.6}$$

The Table 3.2 shows the difference between the different chance models applied to the example in Table 3.1.

Other weighted agreement coefficients

The major weakness of both κ and π is that all disagreements are treated equally. However, some variants consider that not all disagreements are the same. There are

Coefficient	Expected Agreement	Chance-corrected Agreement
S	$2 \times (\frac{1}{2})^2 = 0.5$	$(0.75 - 0.5)/(1 - 0.5) = 0.5$
π	$0.375^2 + 0.625^2 = 0.531$	$(0.75 - 0.531)/(1 - 0.531) \approx 0.467$
κ	$0.35 \times 0.4 + 0.60 \times 0.65 = 0.53$	$(0.75 - 0.53)/(1 - 0.53) \approx 0.468$

Table 3.2: Observed agreement for all the coefficients of 0.75.

two coefficients that differentiate between disagreements, mainly α (Krippendorff, 2004) which supports different magnitudes of disagreement and is based on similar assumptions to those of π, and weighted kappa k_w (Cohen, 1968), a generalization of κ coefficient.

Krippendorff's α: Very similar in assumptions to π, the coefficient α (Krippendorff, 2004) calculates the expected disagreement considering the overall distribution of decisions, not regarding which annotator made those decisions.

To do so, Krippendorff (2004) introduces the notions of **observed disagreement** ($\mathbf{D_o^\alpha}$), and **expected disagreement** ($\mathbf{D_e^\alpha}$), which can be used to calculate α in a way similar to the other coefficients. This representation results form the idea that divergence between disagreement is more relevant than differences between agreement.

To calculate the disagreement between annotators an abstraction is introduced, the notion of distance where $\mathbf{n_{ik}}$ is the number of times the value k was attributed to the item i, and for every ordered pair $k_a, k_b \in K$ there are $\mathbf{n_{ik_a} n_{ik_b}}$ pairs of judgments for item i. The notion of distance $\mathbf{d_{n_{k_a} n_{k_b}}}$ represents the distance between categories (in this case using the number of times items were attributed to that category)n_{k_a} and n_{k_b} and is considered to yield $(n_{k_a} - n_{k_b})^2$ it normally yield the value of 0 if $n_{k_a} = n_{k_b}$ and 1 if $n_{k_a} \neq n_{k_b}$.

These notions allow for all identical category assignments to be counted together. The observed disagreement can be obtained by calculating the sum of distances between categories ($d_{k_j k_l}$) and the number of assignments for each category k (n_{ik_j} and n_{ik_l}) divided by the freedom factor $ic(c-1)$ as shown in equation 3.7.

$$D_o^\alpha = \frac{1}{\mathbf{ic(c-1)}} \sum_{i \in I} \sum_{j=1}^{\mathbf{k}} \sum_{l=1}^{\mathbf{k}} \mathbf{n}_{ik_j} \mathbf{n}_{ik_l} \mathbf{d}_{k_j k_l} \tag{3.7}$$

For the expected disagreement (D^α) similar computation can be made considering the total number of times a category k was assigned to any item by any coder ($\mathbf{n}_k$). Hence the D_e^α can be calculated as shown in equation 3.8.

$$D_e^\alpha = \frac{1}{\mathbf{ic(c-1)}} \sum_{j=1}^{\mathbf{k}} \sum_{l=1}^{\mathbf{k}} \mathbf{n}_{k_j} \mathbf{n}_{k_l} \mathbf{d}_{k_j k_l} \tag{3.8}$$

Since the agreement coefficients are expressed in terms of agreement, the α is computed as the reverse of the reason between observed and expected disagreements, as shown in equation 3.9.

$$\alpha = 1 - \frac{D_o^\alpha}{D^\alpha} \tag{3.9}$$

Cohen's κ_w: Is a weighted variant of Cohen's κ presented in (Cohen, 1968). It is very similar to Krippendorff's α, each pair of categories $k_A, k_B \in K$ is associated to a weight d_{k_A,k_B}. A larger weight denotes more disagreement. Note that there are no constraints on the weights. It is not required that identical categories have a weight of zero. This can be useful in cases where one category is predominant and it is desirable to weight down such category in order to give more relevance to other less predominant categories.

To calculate the disagreement for a particular item i, one should take into account the weight of the pair of categories assigned to it by the two annotators, where the overall observed disagreement is the normalized mean disagreement for all items. Let $k(c_n, i)$ denote the category assigned by annotator c_n to item i. The disagreement for item i is $disagr_i = d_{k(c_1,i)k(c_2,i)}$. The observed disagreement D_o is the mean of $disagr_i$ for all items i, normalized to the interval $[0, 1]$ through division by the maximal weight d_{max}.

$$D_o^{\kappa_w} = \frac{1}{d_max} \frac{1}{i} \sum_{i \in I} disagr_i = \frac{1}{d_max} \frac{1}{i} \sum_{i \in I} d_{k(c_1,i)k(c_2,i)} \tag{3.10}$$

If we consider that all disagreements have the same weight, that is $d_{k_a k_a} = 0$ for all categories k_a and $d_{k_a k_b=1}$ for all categories $k_a \neq k_b$, we can therefore represent the observed disagreement as the complement of the observed agreement calculated for κ: $D_o^{\kappa_w} = 1 - A_o^k$.

The expected disagreement is the amount expected by chance from a probability distribution for each annotator. These individual distributions are estimated by $\hat{P}(k|c)$, the proportion of items assigned to category k by annotator c, that is, the number of such assignments n_{ck} divided by the number of items i.

$$\hat{P}(k|c) = \frac{1}{i} n_{ck} \tag{3.11}$$

The probability of annotator c_1 assigning an item to category k_a and annotator c_2 assigning it to category k_b is the joint probabilities of both assigning them independently, that is, $\hat{P}(k_a|c_1)\hat{P}(k_b|c_2)$. The expected disagreement $D_e^{\kappa_w}$ is the mean of weights for all category pairs, weighted by the probabilities of the category pairs and normalized by dividing it by the maximal weight.

$$D_e^{\kappa_w} = \frac{1}{d_m ax} \sum_{j=1}^{k} \sum_{l=1}^{k} \hat{P}(k_j|c_1)\hat{P}(k_l|c_2)d_{k_j k_l} \tag{3.12}$$

Like the observed disagreement, where all disagreements have the same weight, we can represent the expected disagreement as the complement of the κ expected agreement: $D_e^{\kappa_w} = 1 - A_o^k$

The coefficient κ_w itself is the ratio of observed disagreement to expected disagreement, subtracted by 1 so that it depicts the agreement.

$$\kappa_w = 1 - \frac{D_o}{D_e} \tag{3.13}$$

Both the Krippendorff's α and the Cohen's κ_w are better suited for tasks where the set of categories is limited and known from the start, and where a distance between categories or weight for each category calculation is viable. This however, is not the case for most linguistic annotation tasks.

3.2.2 Cross comparison

Another approach to the calculation of agreement coefficients, consists in applying the metrics used in other areas of NLP for evaluation of the performance of systems or tools, and compare the result of the annotation by the human annotator with those metrics. For instance, in the case of calculating the agreement coefficient in treebanks by humans, the metric used to evaluate the output of parsers against a gold standard treebank are used.

Parseval

The Parseval metric (Black et al., 1991) was conceived for parser evaluation and compares the parser output with the representation deemed to be the correct.

Gold Standard It is the representation (usually a constituency tree) considered correct for a given input. For example, in the case of testing a constituency parser, the gold standard is generally a manually annotated constituency corpora.

In the context of classification tasks, the assignment of a category to an item can be of one of the following four types:

true positive It is when an item is marked as belonging to a category and in the correct result (gold standard) it belongs to that category.

false positive It is when an item is marked as belonging to a category but in the correct result it does not belong to that category.

true negative It is when an item is marked as not belonging to a category and in the correct result it does not belong to that category.

false negative It is when an item is marked as not belonging to a category when in the correct result it belongs to that category.

Precision In the field of parser evaluation, it is the proportion of correct constituents (yield) in the resulting representation, when compared to the gold standard.

$$\text{precision } (P) = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{3.14}$$

Recall In the field of parser evaluation, it is the proportion of correct constituents (yield) in the resulting representation, when compared to the gold standard.

$$\text{recall } (R) = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{3.15}$$

To better illustrate this metric, lets consider the sentence "O João viu a Maria com os binóculos." and the two following syntactic tree representations Figure 3.1 considered the gold standard and Figure 3.2 the representation to be evaluated.

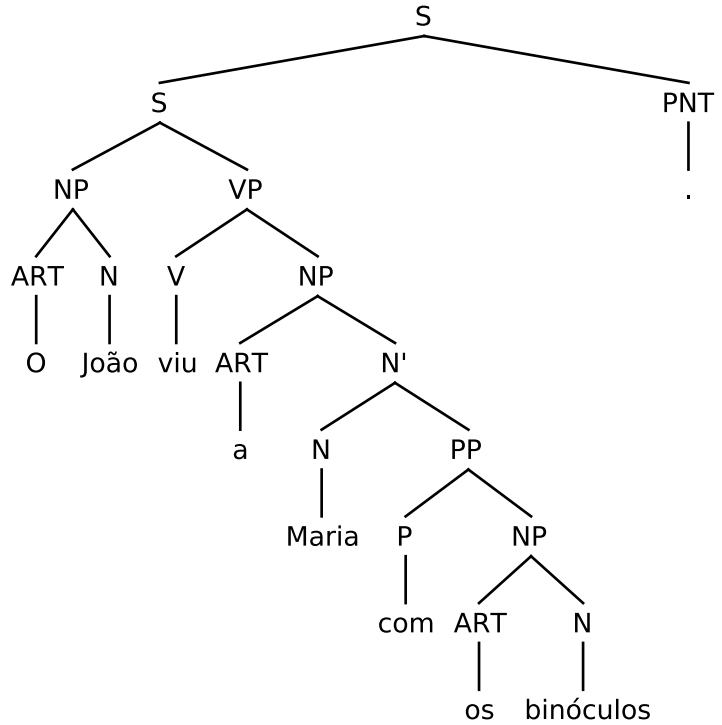


Figure 3.1: Tree representation for the sentence "O João viu a Maria com os binóculos." considered the gold standard.

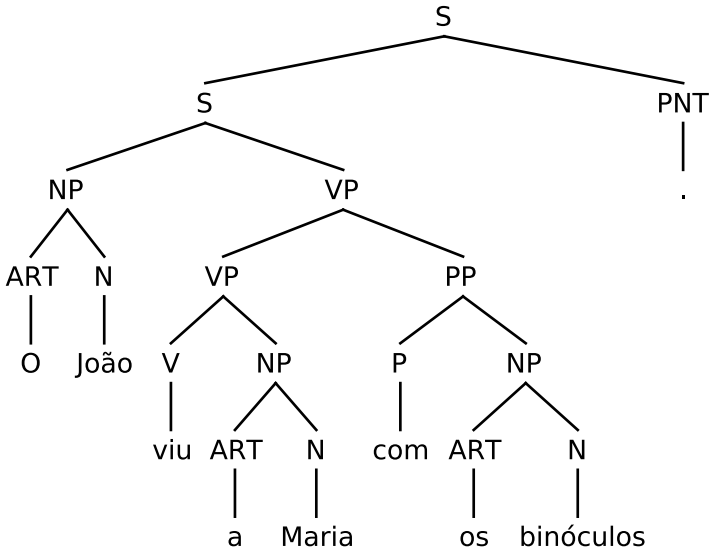


Figure 3.2: Tree representation for the sentence "O João viu a Maria com os binóculos." to be evaluated.

To better understand the calculation of the Parseval metric it is necessary to understand what the span of the constituents is, it is the coverage of each constituent, for example the constituent `ART` span starts at token 1 and ends at token 1 since the brackets surrounding `ART` only cover the first token.

In order to easily calculate the Parseval metric, a list of triples containing the beginning and ending of the constituent coverage and the constituent are formed for each tree, and are then compared.

For the trees in Figure 3.1 and 3.2 the triples lists are respectively represented in Figure 3.3.

Figure 3.1 triples	**Figure 3.2 triples**
(1, 1, 'ART')	(1, 1, 'ART')
(1, 2, 'NP')	(1, 2, 'NP')
(1, 8, 'S')	(1, 8, 'S')
(1, 9, 'S')	(1, 9, 'S')
(2, 2, 'N')	(2, 2, 'N')
(3, 3, 'V')	(3, 3, 'V')
(3, 5, 'VP')	(3, 8, 'VP')
(3, 8, 'VP')	(4, 4, 'ART')
(4, 4, 'ART')	(4, 8, 'NP')
(4, 5, 'NP')	(5, 5, 'N')
(5, 5, 'N')	(5, 8, "N'")
(6, 6, 'P')	(6, 6, 'P')
(6, 8, 'PP')	(6, 8, 'PP')
(7, 7, 'ART')	(7, 7, 'ART')
(7, 8, 'NP')	(7, 8, 'NP')
(8, 8, 'N')	(8, 8, 'N')
(9, 9, 'PNT')	(9, 9, 'PNT')

Figure 3.3: Parseval span triples for two representations of the sentence "O João viu a Maria com os binóculos."

With this values it is easy to calculate the Precision and Recall values.

$$\text{Precision } (P) = \frac{\text{\# Correct Constituents}}{\text{\# Constituents in parser output}} = \frac{15}{17} = 0.88$$

$$\text{Recall } (R) = \frac{\text{\# Correct Constituents}}{\text{\# Constituents in gold standard}} = \frac{15}{17} = 0.88$$

The Parseval metric result consists in this two values. However, in order to improve the accuracy of this values the F-measure or F-score is calculated over this values.

F-Score is the harmonic mean between precision and recall. It is often used in the field of information retrieval and question answering. It can be calculated the following way:

$$\text{F-Score } (F) = \frac{2PR}{P+R} \tag{3.16}$$

F-Score for Inter-annotator agreement

This coefficient is mentioned by Brants (2000) when dealing with the NEGRA corpus annotation task. Some adaptations to the F-Score above are made for the inter-annotation agreement measured by F-Score when 2 annotation versions A_1 and A_2 are considered (i.e. two annotators).

$$\text{precision}(X,Y) = \frac{\text{number of identical nodes in } A_1 \text{ and } A_2}{\text{number of nodes in } A_2} \tag{3.17}$$

$$\text{recall}(X,Y) = \frac{\text{number of identical nodes in } A_1 \text{ and } A_2}{\text{number of nodes in } A_1} \tag{3.18}$$

$$\text{F-Score } (F) = \frac{2PR}{P+R} \tag{3.19}$$

Note that $\text{recall}(X,Y) = \text{precision}(X,Y)$. Since there is no correct version when comparing the two annotators, the F-Score is considered to be the most appropriate of this measures for inter-annotator agreement.

Leaf-Ancestor

The Parseval metric is deemed to penalize too much some attachment errors and to be too sensitive to the annotation scheme (Rehbein and van Genabith, 2007). Accordingly, the Leaf-Ancestor metric has started to be used to evaluate parsers.

The Leaf-Ancestor metric was introduced by Sampson and Babarczy (2003) and like the Parseval metric it was developed to evaluate the output of parsers. The metric compares the paths between the terminal nodes and the root node of the parse trees. This paths consists on the labels between the terminal node and the

root node. The path for each terminal node of the parse tree and the gold standard are compared using the Levenshtein distance.[1]

The score of the whole tree is the average of the values for all terminal nodes in the tree.

All this can be better understood with an example. Using as gold standard the representation in Figure 3.1 and as parse analysis the representation in Figure 3.2, the leaf ancestor calculates the distances for each leaf, as it is illustrated by Figure 3.4 and yields an average distance of 0.914.

Distance	**Leaf**	**Parse Analysis String** : **Gold Standard String**	
1.000	O	ART] NP S [S	: ART] NP S [S
1.000	João	[N NP] S S	: [N NP] S S
0.923	viu	V] VP [VP S S	: V] [VP S S
0.933	a	ART] [NP VP VP S S	: ART] [NP VP S S
0.719	Maria	[N NP VP] VP S S	: N] [N' NP VP S S
0.875	com	P] [PP VP S S	: P] [PP N' NP VP S S
0.889	os	ART] [NP PP VP S S	: ART] [NP PP N' NP VP S S
0.889	binóculos	[N NP PP VP S] S	: [N NP PP N' NP VP S] S
1.000	.	[PNT S]	: [PNT S]

Figure 3.4: Leaf Ancestor distances for the sentence "O João viu a Maria com os binóculos." with an average distance or leaf ancestor rating of 0.941.

Elementary Dependency Match

The Elementary Dependency Match (EDM) was introduced by Dridan (2009) and takes into account the representation enconded in terms of the MRS semantic representation formalism (Copestake et al., 2005). This semantic representation is transformed into triples which are then compared with the gold analysis version. Missing triples are replaced with empty triples.

The triples can be of one of three types:

$$\begin{array}{lccc} NAMES: & span_i & NAME & relation_i \\ ARGS: & span_i & role_j & span_k \\ PROPS: & span_i & property_j & value_j \end{array}$$

An example can better illustrate the metric, taking as reference the sentence "O Manuel foi apenas à loja.". The gold standard has the triples in Figure 3.5 and the "parse" triples are shown in Figure 3.6.

The evaluation tool outputs the list of triples items with a **diff** notation indicating the differences found. The > indicate in the gold standard, and < in the parsed version.

The output for the sentence previously presented is shown in Figure 3.7.

[1]The Levenshtein distance between two strings can be defined by the minimum number of operations of the type insertion, deletion or substitution of a single character to transform one string into the other.

_ir_v_dir<9:11>	ARG1	named
_ir_v_dir<9:11>	ARG2	_loja_n<24:27>
_apenas_a<13:18>	ARG1	_ir_v_dir<9:11>
_o_q<0:0>	ARG0	_loja_n<24:27>
_ir_v_dir<9:11>	ELLIPTICAL-PUNCT	bool
_ir_v_dir<9:11>	SF	proposition-or-question
_ir_v_dir<9:11>	E.TENSE	pretérito-perfeito
_ir_v_dir<9:11>	E.MOOD	indicativo
_loja_n<24:27>	PERSON	3rd
_loja_n<24:27>	NUMBER	singular
_loja_n<24:27>	GENDER	feminine

Figure 3.5: EDM gold standard triples for the sentence “O Manuel foi apenas à loja.”.

_ir_v_dir<9:11>	ARG1	named
_ir_v_dir<9:11>	ARG2	_loja_n<24:27>
_apenas_a<13:18>	ARG1	_ir_v_dir<9:11>
_o_q<0:0>	ARG0	_loja_n<24:27>
_ir_v_dir<9:11>	ELLIPTICAL-PUNCT	bool
_ir_v_dir<9:11>	SF	proposition-or-question
_ir_v_dir<9:11>	E.TENSE	pretérito-perfeito
_ir_v_dir<9:11>	E.MOOD	indicativo
_loja_n<24:27>	PERSON	3rd
_loja_n<24:27>	NUMBER	singular
_loja_n<24:27>	GENDER	feminine

Figure 3.6: EDM parse triples for the sentence “O Manuel foi apenas à loja.”.

9.gz:>	”o”	<0:0>	:ARG0:”loja”	<24:27>
9.gz:>	”o”	<0:0>	:NAMES:_o_q	
9.gz:>	”foi”	<9:11>	:ARG1:named	
9.gz:>	”foi”	<9:11>	:ARG2:”loja”	<24:27>
9.gz:>	”foi”	<9:11>	:E.MOOD:indicativo	
9.gz:>	”foi”	<9:11>	:E.TENSE:pretérito-perfeito	
9.gz:>	”foi”	<9:11>	:ELLIPTICAL-PUNCT:bool	
9.gz:>	”foi”	<9:11>	:NAMES:_ir_v_dir	
9.gz:>	”foi”	<9:11>	:SF:proposition-or-question	
9.gz:>	”apenas”	<13:18>	:ARG1:”foi”	<9:11>
9.gz:>	”apenas”	<13:18>	:NAMES:_apenas_q	
9.gz:<	”apenas”	<13:18>	:NAMES:_apenas_a	
9.gz:>	”loja”	<24:27>	:GENDER:feminine	
9.gz:>	”loja”	<24:27>	:NAMES:_loja_n	
9.gz:>	”loja”	<24:27>	:NUMBER:singular	
9.gz:>	”loja”	<24:27>	:PERSON:3rd	

Figure 3.7: EDM output triples for the sentence “O Manuel foi apenas à loja.” with the *diff* notation.

For this example, the EDM scores a precision of $15/16 = 0.938$ a recall of $15/16 = 0.938$ and an f-score of $\frac{2\times0.938\times0.938}{0.938+0.938} = 0.938$

This metric is tuned for the evaluation of deep linguistic parsers and requires the existence of a gold analysis. It also considers discriminants that can be added by

the LOGON tool automatically as direct result of manual discriminant assignment. The focus of our work however is to find a metric for inter-annotation evaluation used to quantify the quality of the production of such golden analysis when there is no gold standard and only two equaly weighted human analysis to compare.

3.3 Summary

Cohen's κ coefficient and further adaptations are highly valuable. The adaptations made to the metrics aim at better approaches to estimating the annotators behavior when real annotation versus random annotations are considered.

Nevertheless, very little focus is given to the granularity of the annotation process itself. This is mainly due to the fact that the type of annotation being assessed by most of the studies in the literature is quite shallow. In most cases, it is either POS tagging or identification of specific phenomena, and these cases require no more granularity then the one already available.

When it comes to more complex categories, like for instance deep linguistic grammatical, representations the F-Score measure gives a good idea on how close the two annotations are. However, this metric is only appropriated to compare parser outputs which are usually in tree format. This only allows the comparison of one of the vistas of the LXDeepGramBank and not the whole information. This leaves a great deal of information without comparison, such as some deep grammatical information, leading to ignore part of the information the annotators have to decide about.

The EDM is an enhanced metric for deep linguistic grammars, however it is optimized for such task, and leaves some room for granularity improvement, specially at the level of distinguishing automated and manual discriminant assignment.

Chapter 4

Agreement Coefficient for Deep Linguistic Parsing

Nearly all corpora annotation efforts predating Carletta's (1996) paper do not mention any reliability values, metrics or coefficients. Even more recent initiatives only mention observed agreement like the PropBank (Palmer et al., 2005, 2007) or choose only to evaluate the shallower representation of the annotated dataset like the NEGRA corpora.

Likewise the observed agreement lacks the necessary strength to evaluate corpora annotated with deep grammatical information similarity. Additionally, the aforementioned F-Score approach, which only takes into account the constituency tree representation of the deep linguistic annotated sentences, also lacks the necessary granularity to assess the reliability of deep linguistic annotation.

4.1 Desired Granularity

The different approaches described in Chapter 3.2 focus almost entirely on mitigating the issue of random decision, and largely ignore the lack of granularity of these methods. This can be justified by the level of complexity of the annotation tasks the metrics were developed for and used with.

The lack of granularity to deal with complex tasks such as the correct evaluation of corpora annotated with deep linguistic information is easy to illustrate with a small example.

Consider two parse trees, T_{sC_1} and T_{sC_2} assigned by two annotators (c_1 and c_2 respectively) to sentence s. These two trees may differ only in terms of the attachment of a given PP. This means that, out of the decisions the two annotators had to make to obtain those trees that should annotate s, the two annotators disagreed on only one discriminant, namely on where to attach the PP. However, the difference in that PP attachment in the two trees may result in several differences in terms of phrase boundaries and tree representations.

On an exact match approach, this would mean complete disagreement and even in a more granular approach, as the ones used to evaluate the parsers output mentioned in the previous chapter, would yield low levels of agreement.

This is, without any doubt, a very unfair result, both for the annotators, who only differed in one of many discriminants, and for the annotation outcome that ends up being ranked unfairly by low levels of reliability. It is interesting to note that this exact problem was identified for the metrics for parsers evaluation (Lin, 1998).

A metric employed in the evaluation of corpora annotated with deep linguistic information should have enough granularity to disentangle such cases at the appropriate level.

The human component of the task of annotating a corpora with deep linguistic information is in fact a task of disambiguation between many possible grammatical representations for the sentences of a corpora, and not actually annotation of sentences, or trees with tags or categories. As it was mentioned in Chapter 2.6.1, the task consists in taking into consideration the semantic discriminants in the format of binary discriminants that can be marked as 'yes' or 'no' for a given sentence.

For this task, the desired granularity would consist in taking into account the set of discriminants each annotator considered for each sentence and comparing them one by one. Under this scheme, the example above about PP attachment would only differ on a single discriminant from the whole set of options selected during the disambiguation of the given sentence, making this evaluation much more accurate and truly granular.

4.2 Notions and Notation

There are some specific notations for the metrics argued for in this document, as well as some notions that might differ from the ones presented in other publications.

Specifically, there is a set of **sentences** S or corpus, and each sentence $s \in S$ has a set of semantic **discriminants**. This set is called the set of **markables** of sentence s. The act of assigning a *yes* or *no* value to one of this discriminants is referred to as an **option**. The subset containing only the discriminants assigned with the value *yes* is called the **Y-options** set. The act of either accepting a grammatical representation for a sentence s because a set of discriminants was found that results in the grammatical representation considered correct by the annotator or the rejection because no such set could be found, is called a **decision**.

To better illustrate these notions, in Figure 4.1, a screenshot of the LOGON interface for the sentence "Foi pura coincidência" (It was pure coincidence.) has the respective set of semantic discriminants or markables on the right side of the

divided window frame. All the available discriminants can have a value of *yes* or *no* option assigned to them, thus trimming the possible grammatical representations, which in this case are rendered in a simplified vista as constituency trees in the left side of the graphical interface for the sake of a first easy grasping by annotators.

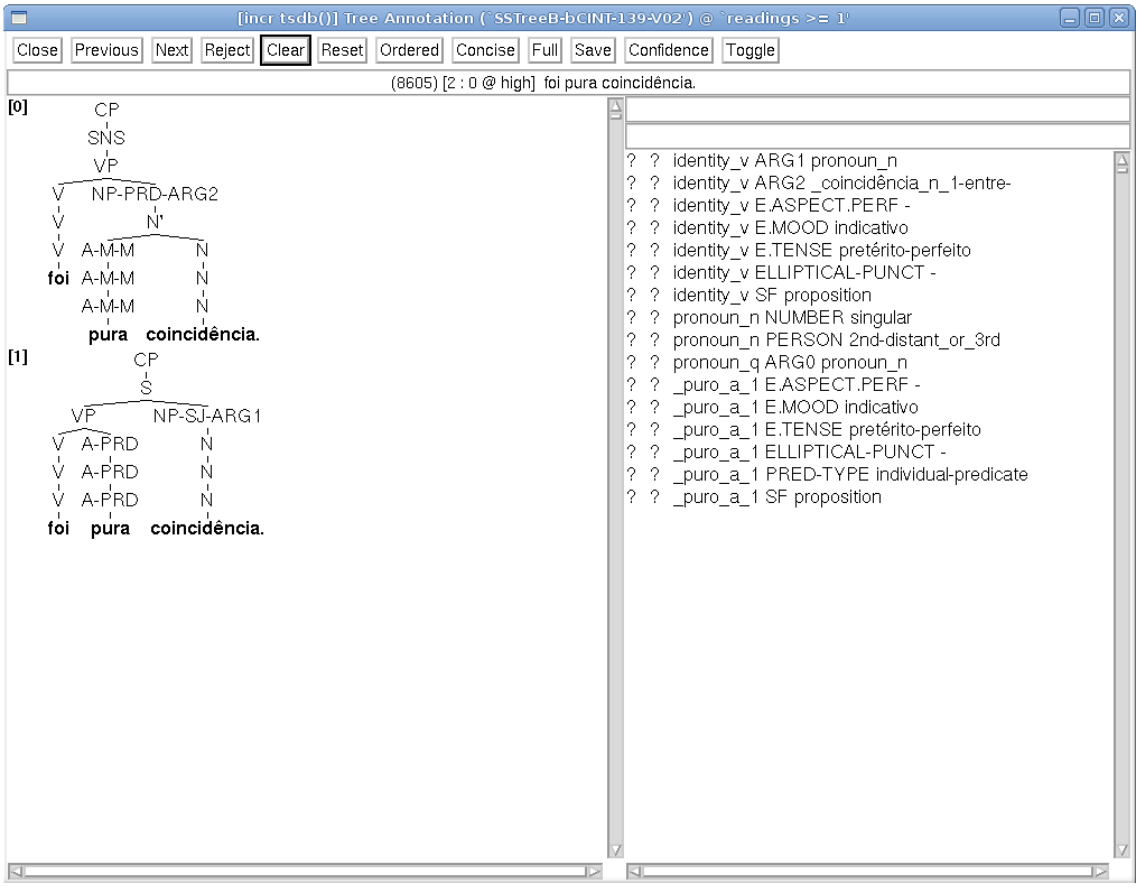


Figure 4.1: The LOGON interface for the sentence "Foi pura coincidência." (It was pure coincidence.). The possible semantic discriminants for the sentence are visible on the right side of the graphical interface.

More details on the meaning of the discriminants and the impact of assigning *yes* or *no* values to them will be discussed in detail in the next chapter.

Some of the notations previously introduced on Chapter 3.2 also have slight adjustments in this chapter.

That is the case of the notations for the agreement. In the previous chapter, the notation A^X referred to the agreement for coefficient X. For the sake of convenience, however, in the present chapter A^X refers to the agreement for the X dataset, and the notation for expected and observed agreement are respectively A_e^X and A_o^X.

The notation of the coefficient K undergone the same kind of adjustment. The notation K_Y, denotes the agreement coefficient Y-option kappa, and the K_Y^x notation, which refers to the coefficient K_Y when calculated for the sentence x.

4.3 Agreement

For an agreement metric to consistently and efficiently evaluate the process of corpora annotation with deep linguistic information, one must take the annotation process into account, since it is exactly the reliability of the outcome of this process that is being assessed.

4.3.1 Observed Agreement

What is proposed here is that the observed agreement is measured as the proportion of discriminants on which the annotators agree from the total set of available discriminants.

For example, considering a sentence s with a set D_s of 5 options, if two annotators agree on 4 of these 5 available options, it can be said that the **observed agreement** for sentence s is the proportion of discriminants on which both annotators agreed upon, in this case, $A_o^s = 0.8$ (on a scale where 1 is total agreement and 0 total disagreement) when agr_s is the number of coincident options, this can be rendered as:

$$A_o^s = \frac{agr_s}{|D_s|} \tag{4.1}$$

This approach to determining the observed agreement between annotators allows for the highest possible granularity with the used annotation scheme since the considered options are atomic units, that is, the smallest possible units in the decision process. In contrast to other approaches, it is possible for annotators not to mark the exact same set of discriminants with 'yes' (full agreement), but to agree on a subset of discriminant and still achieve a certain level of agreement (partial agreement).

4.3.2 Expected Agreement

Since each option is intrinsically related to the words and the sentence it is associated with, the options are in a large majority unique, that is, each option typically only exists for one sentence in the entire corpora. Since most distributions used to estimate the annotator behavior by chance depends on previous knowledge or data for a given set of markables, using any distribution other than uniform distribution for the expected agreement by chance alone is thus not viable here.

This being said, it can be assumed that if the annotators were trying to mark the entire set of available options for any given sentence s by chance alone, the result would be the result of a uniform distribution of binary decisions. This process can

be compared to the flipping a coin, and the expected result is the same, hence the expected agreement (A_e^s) is 0.5.

4.4 The coefficient

The calculation of the agreement coefficient is simple once the expected agreement and observed agreement are calculated. The agreement for sentence s of the corpora can be calculated by taking into account the observed agreement and weighting it with the expected agreement for s. This coefficient can be called Y-option Kappa or K_Y and for sentence s can be calculated by:

$$K_Y^s = \frac{A_o^s - A_e^s}{1 - A_e^s} \tag{4.2}$$

The A_o^s and A_e^s are respectively the observed agreement and expected agreement for sentence s. The agreement for the entire corpora, or K_Y^S is the ratio between the sum of the agreement for all the sentences of the corpora and the number of sentences in the corpora.

$$K_Y^S = \frac{\sum_{s \in S}^{|S|} K^s}{|S|} \tag{4.3}$$

4.5 Summary

The metric proposed in this document offers the maximum granularity available for the task at hand, ensuring the evaluation of the corpora annotation in a detail that allows a distinction between analysis that can diverge only in one discriminant.

This allows for a detailed analysis and evaluation of corpora annotated with deep linguistic information, allowing for a more precise evaluation of annotation that would otherwise have unfair evaluations thanks to their coarse granularity.

Chapter 5

Experimental Assessment

In the previous chapter, the theoretical definition of the agreement metric Y-Option Kappa (K_Y) was put forward. The actual interest of this metric is for it to be applied to practical cases, as in the development of the LXDeepGramBank.

The present chapter makes a detailed description of the annotation process, the tools used for this task, how the data is saved, recovered and analyzed for the creation of this corpus. It also discusses the permitted adaptations of the theoretical coefficient that turns out to be convenient for it to cope with the specific corpus and data format used by those tools.

5.1 LOGON treebanking environment

LOGON (Lønning et al., 2004) is an integrated package for diagnostics, evaluation and benchmarking in practical grammar engineering. It is almost fully implemented in Common-Lisp with a Tcl/Tk widget graphical interface.

Since the main focus of the developer of this tool was the development of computational grammars, most of the package features are related to the test and enhancement of the grammar itself, and not so much with the corpora annotation task. It is however the only application that allows dynamic annotation, which means the grammatical representations and the implications from the annotator decisions are computed dynamically.

The LOGON allows the user to gather some metrics used to control the computational grammar, such as the parse time, memory used, number of nodes or parse results and coverage. Our specific task is focused almost totally on the annotation interface as a mean to assist the human annotators on their annotation task.

5.1.1 Structure and Interface

Since most of LOGON tools are entirely in mostly Common-Lisp, it is executed in the emacs environment. The user interface is divided into 4 main function windows.

Output window is where the system output is shown, and loading messages, error reports and process information are displayed in the form of text messages. An example of this area can be seen in Figure 5.1. Since in this case LOGON is executed in the emacs environment the output window is the emacs environment.

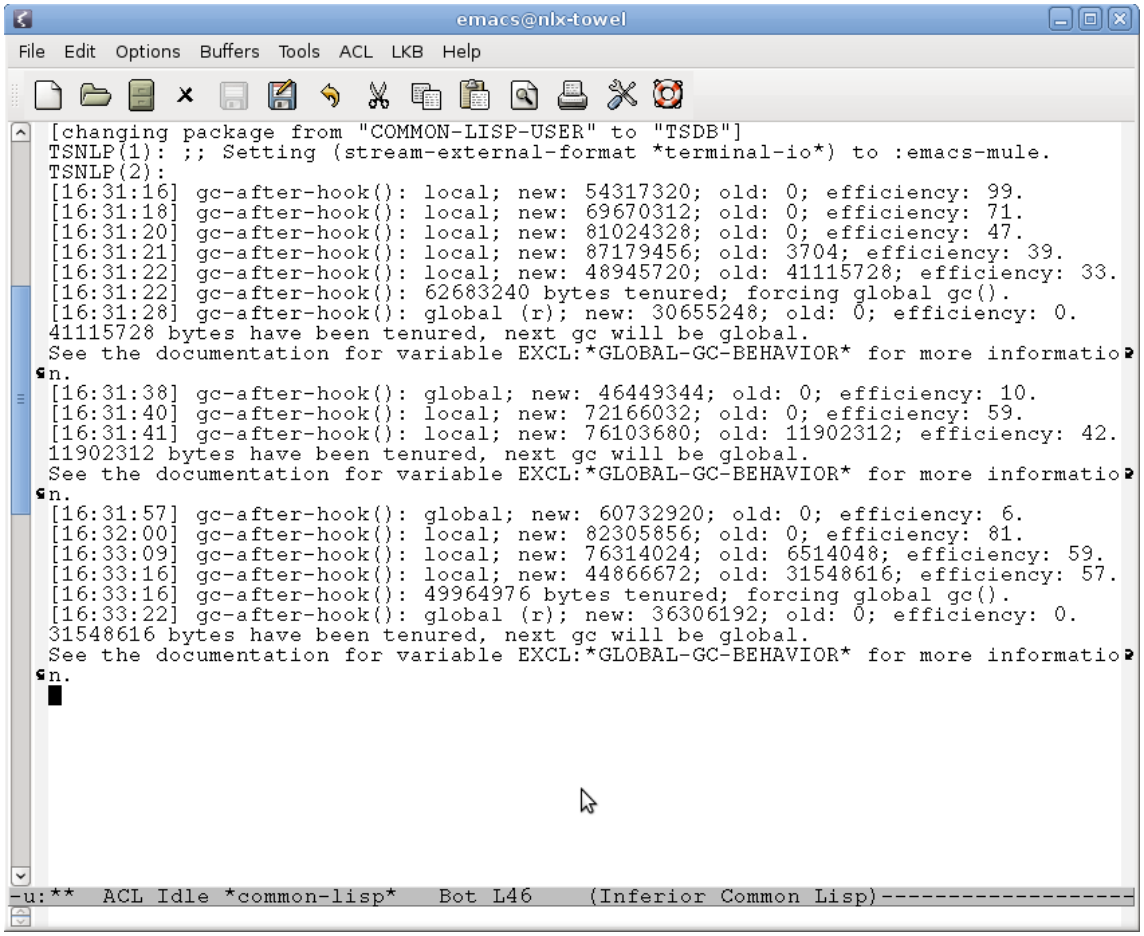


Figure 5.1: The LOGON(emacs) output window in a emacs console.

LKB is the application area (Figure 5.2) where the user can interact with the grammar. It allows the user to load a grammar (HPSG grammar and PET parser precompiled), and it also allows for queries over the loaded grammar, for example one can see the type hierarchy or parse a single or multiple sentences.

[incr tsdb()] podium is the interface part reserved to the corpora being tested or annotated (Figure 5.3). It allows the user to interact with the corpora, add and remove suites,[1] tweak options like the maximum number of nodes or parses to be generated, and it also allows to have access to coverage metrics.

[1]Suite is a dataset fraction in which a corpus is divided to make the processing more efficient.

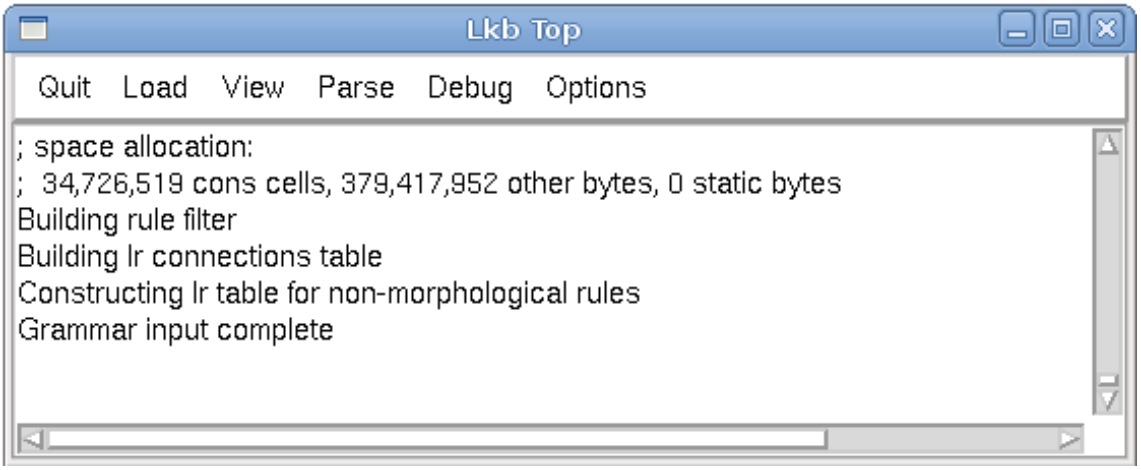


Figure 5.2: The LKB interface window.

The podium is also where the corpora can be extracted suite by suite into the typical formats such as Penn treebank, AVM, and Redwood.

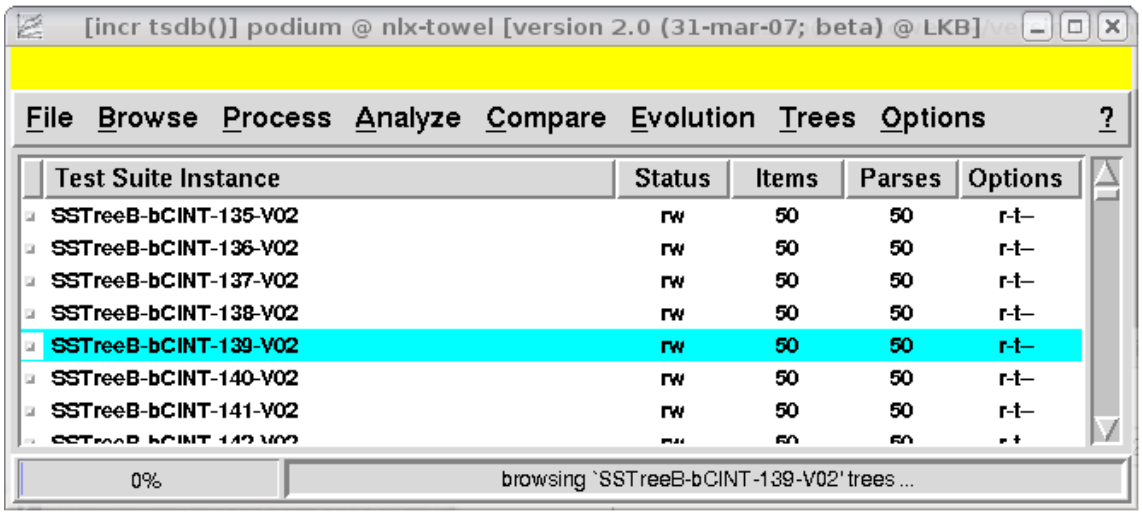


Figure 5.3: The Podium interface window.

[incr tsdb()] Tree Annotation is the environment where the user can annotate the corpora by selecting options from the available set of discriminants. The possible parse trees with the available and assigned discriminants are shown on the left side of the window and the unassigned discriminants are listed on the right side of the window.

The interface is very simple. The sentences are displayed sequentially when the next or previous button are pressed. All decisions can be reset with the "Reset" key, and saved to keep the decisions made.

The tree annotation interface for the sentence "Foi pura coincidência." ("It was pure coincidence.") is represented in Figure 5.4.

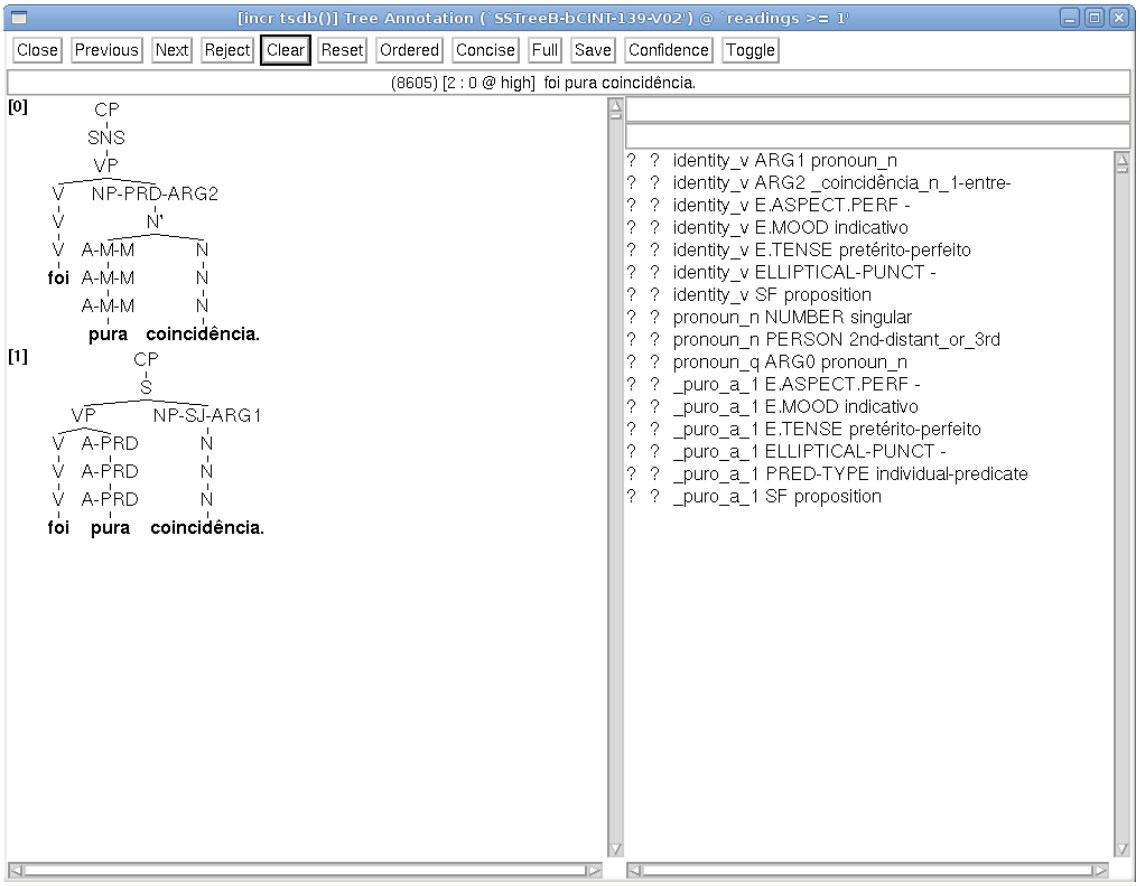


Figure 5.4: The Tree Annotation interface window.

5.1.2 Tree Annotation Process

Consider the case of Figure 5.4 as an example. As mentioned earlier in this document, the annotation process consists in human annotators deciding the discriminants as belonging (*yes*) or not (*no*) to the grammatical representation considered as the most correct for the given sentence by the human annotator.

When the process starts, the annotator has all the available grammatical representations for the sentence (a.k.a. parse trees, or parses) on the left side and all the possible discriminants on the right side.

In this case, there are 16 discriminants available for this sentence, and two possible grammatical representations in tree format for easier interpretation by humans.

It is important to point out that for efficiency reasons, the [incr tsdb()] only presents the discriminants which induce divergence for one or more grammatical representations, hence, discriminants that belong to all representations are not shown or registered. For example, for the sentence in the Figure 5.4, options about any of the available discriminants will cause at least one of the sentence parses to be excluded. For example, if the annotator decides the entry *identity_v ARG1 pronoun_n* which indicates the identity reading of the verb with the (null) pronoun as it is ARG1,

(in this case the verb 'foi' has an ARG1 which is not present in the sentence), thus the grammatical representation that identifies the word 'coincidência' as ARG1 is eliminated, resulting in only one valid grammatical representation being left for the annotator to accept or reject, the one with the null subject (SNS).

This optimizations makes it possible that not all the discriminants are shown to the annotator, nor considered in the annotation process. This reduces the number of possible options from all the grammatical options to those that can reduce the parse forest dimension.

It is also important to note that the [incr tsdb()] does not use stored grammatical analysis for the annotation process. All the discriminants and grammatical analysis (parses) are generated 'on the fly' using the grammar while the annotators interact with the tool. This means that the process is very heavy and can result in some sentences or analysis being lost by the hardware or the memory failing to fulfill the requirements for the 'on the fly' grammatical processing.

5.1.3 Summary

The LOGON is a good collection of tools for grammar developers to gather statistics about the grammar and the parsing process, and has an acceptable graphical interface that makes the selection and assignment of discriminants easier for the humans annotators. However it focus on the grammar engineering needs, considering only metrics like coverage, number of parses or time to parse and neglecting important information for the evaluation of the annotation process, such as storing the entire discriminant set.

5.2 The [incr tsdb()] log files

The [incr tsdb()] tool generates a series of logs that contain not only statistical information regarding the parsing of the corpora, but also the necessary information to generate the parse of the sentences of a suite. These parses correspond to the ones selected by the human annotators as correct for each sentence of the given suite.

Some of these log files are only used in specific situations, others are used in the whole process.

These logs are the only information about the annotation process as well as about its result. Since this tool has a special focus on improving and benchmarking grammar engineering, it only records the minimal information regarding the annotation process. Also, the logs are not meant to be read by humans and their encoding can be of hard interpretation by humans.

There are many different log files, some are only used for grammar debugging and are not used in the annotation task, thus resulting in empty files. Others are

```
8605@5@1@6@identity_v ARG2 _coincidência_n_1-entre-@@0@0@5-may-2009 11:25
8605@5@4@6@_puro_a_1 E.ASPECT.PERF -@@v0@0@5-may-2009 11:25
8605@5@4@6@_puro_a_1 E.MOOD indicativo@@0@0@5-may-2009 11:25
8605@5@4@6@_puro_a_1 E.TENSE pretérito-perfeito@@0@0@5-may-2009 11:25
8605@5@4@6@_puro_a_1 ELLIPTICAL-PUNCT -@@0@0@5-may-2009 11:25
8605@5@4@6@_puro_a_1 PRED-TYPE individual-predicate@@0@0@5-may-2009 11:25
8605@5@4@6@_puro_a_1 SF proposition@@0@0@5-may-2009 11:25
```

Figure 5.5: Content of the [incr tsdb()] log file `decision` for the sentence "Foi pura coincidência." ("It was pure coincidence.").

fundamental for the analysis and result comparison. There is a set of log files for each suite. The relevant files are: `decision`, `result`, `parse`, and `preference`.

The following sections describe these log files in more detail.

5.2.1 The `decision` log file

The decision log file contains the discriminants accepted as well as some of the rejected (the strictly necessary to generate the correct grammatical representation) for each sentence of the suite.

The different fields of information contained in the lines of the decision log file are separated by @'s. Taking as reference the last line of Figure 5.5 they are:

- sentence identification: the name or number of the sentence, in this case '`8605`'.
- number of the revision: each sentence can be annotated and reviewed multiple times by an annotator, in this case this is the 5th revision.
- state of the discriminant: it can be an integer from `1` to `4` or `-1` in case the sentence was rejected as having no valid parses.
  - `1` - Indicates the annotator marked the discriminant as *yes* manually.
  - `2` - Indicates the annotator marked the discriminant as *no* manually.
  - `3` - Indicates the discriminant was automatically marked as *yes* by implication of some other manual decision.
  - `4` - Indicates the discriminant was automatically marked as *no* by implication of some other manual decision.

    These cases are better explain in Section 5.2.1.

  In the case of the example, the first discriminant was marked as *yes* by the annotator and the other discriminants were automatically marked as *no* because the parse they belonged to is no longer available.

- discriminant key: this is the text representation of the discriminant, in this case, '*_puro_a_1 SF proposition*'.

- the date of the annotation: which was '5-may-2009 11:25'.

The fields left unmentioned are not used in the annotation or parsing process of the LXDeepBank, hence are left empty or have no relevant information.

It is also important to point out that in case of rejection of the sentence, this means that there were possible grammatical representations for the sentence, but none was considered to be correct by the annotator. In such a case, special entry is introduced in the decision log file under the format:

```
8620@3@-1@5@@@@0@19@28-apr-2009 10:59
```

Which means that sentence 8620 was rejected. The rejection is denoted by the lack of discriminant information and the third field value (`-1`).

The discriminants have a defined order in the discriminant set (this order is evident when the annotator accepts the parse tree directly since all discriminants are stored). Taking aside the ones omitted by [incr tsdb()] in the moment of storage, all the discriminants are stored in the same order in which they are presented in the graphical interface, except for the discriminants manually marked as *yes* which always precede the automatic decisions caused by this decision. This can be perceived by comparing the decision log file output presented in Figure 5.5 and the discriminants list in Figure 5.4. It is evident that the order of the discriminants is the same, except for the discriminants omitted by the [incr tsdb()] when storing the log information.

Discriminant storing in the [incr tsdb()] `decision` log file

The information stored in the [incr tsdb()] `decision` log files depends on how the final parse was achieved.

`Annotator selects a parse directly from the tree`

When an annotator selects a sentence directly by marking the tree as *yes* as (shown in Figure 5.6), all the discriminants are stored in the `decision` log file. The discriminants that belong to the selected parse are stored with value `3`. The ones that do not belong to that parse are stored with the value `4`. The content of the `decision` log file for this case is shown in Figure 5.7.

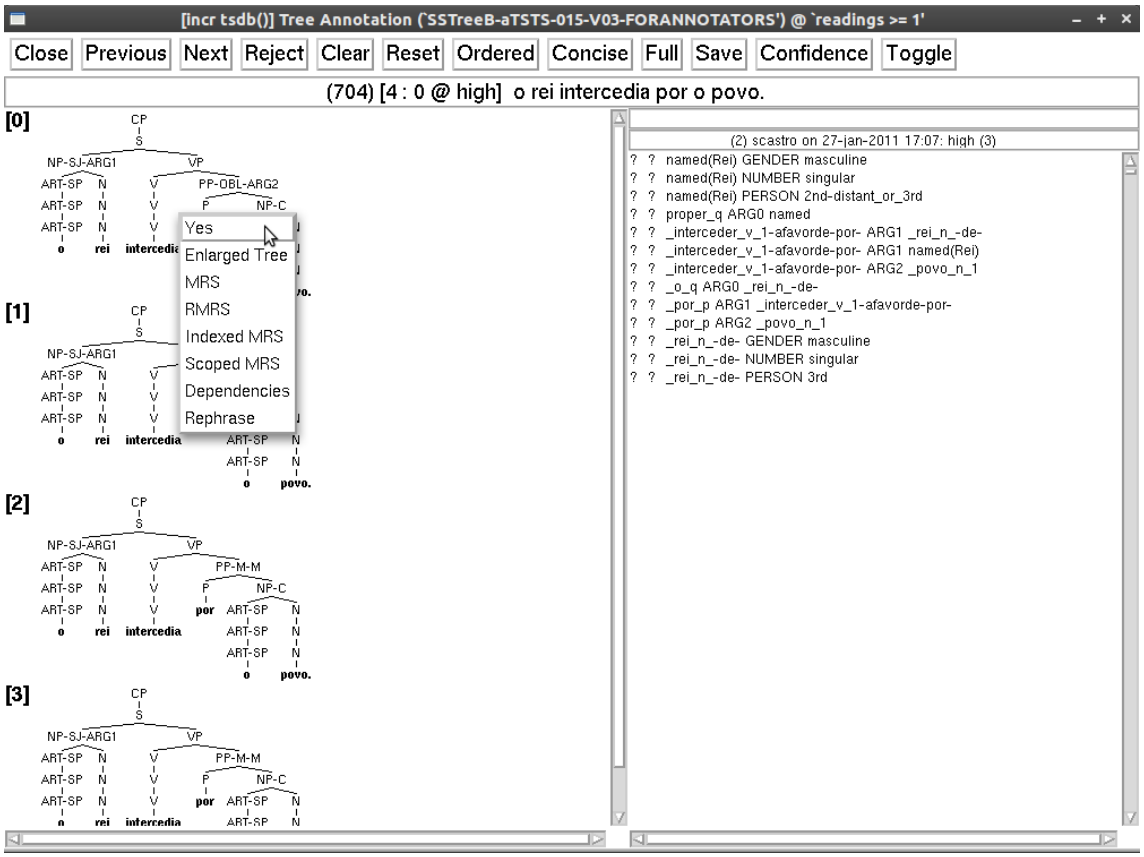


Figure 5.6: Direct acceptance of a parse tree in the LOGON annotation interface.

```
704@2@3@6@named(Rei) GENDER masculine@@0@0@28-apr-2009 10:59
704@2@3@6@named(Rei) NUMBER singular@@0@0@28-apr-2009 10:59
704@2@3@6@named(Rei) PERSON 2nd-distant_or_3rd@@0@0@28-apr-2009 10:59
704@2@3@6@proper_q ARG0 named@@0@0@28-apr-2009 10:59
704@2@4@6@_interceder_v_1-afavorde-por- ARG1 _rei_n_-de-@@0@0@28-apr-2009 10:59
704@2@3@6@_interceder_v_1-afavorde-por- ARG1 named(Rei)@@0@0@28-apr-2009 10:59
704@2@3@6@_interceder_v_1-afavorde-por- ARG2 _povo_n_1@@0@0@28-apr-2009 10:59
704@2@4@6@_o_q ARG0 _rei_n_-de-@@0@0@28-apr-2009 10:59
704@2@4@6@_por_p ARG1 _interceder_v_1-afavorde-por-@@0@0@28-apr-2009 10:59
704@2@4@6@_por_p ARG2 _povo_n_1@@0@0@28-apr-2009 10:59
704@2@4@6@_rei_n_-de- GENDER masculine@@0@0@28-apr-2009 10:59
704@2@4@6@_rei_n_-de- NUMBER singular@@0@0@28-apr-2009 10:59
704@2@4@6@_rei_n_-de- PERSON 3rd@@0@0@28-apr-2009 10:59
```

Figure 5.7: Content of the [incr tsdb()] log file `decision` for sentence 704 when the annotator selects a parse tree directly. The discriminants with a third value of 3 contributed to the selected parse tree, the ones with 4 do not.

```
704@5@1@6@named(Rei) GENDER masculine@@0@0@28-apr-2009 10:59
704@5@4@6@_interceder_v_1-afavorde-por- ARG1 _rei_n_-de-@@0@0@28-apr-2009 10:59
704@5@4@6@_o_q ARG0 _rei_n_-de-@@0@0@28-apr-2009 10:59
704@5@4@6@_rei_n_-de- GENDER masculine@@0@0@28-apr-2009 10:59
704@5@4@6@_rei_n_-de- NUMBER singular@@0@0@28-apr-2009 10:59
704@5@4@6@_rei_n_-de- PERSON 3rd@@0@0@28-apr-2009 10:59
```

Figure 5.8: Excerpt of the [incr tsdb()] `decision` file for the case in which the annotator marked the '*named(Rei) GENDER masculine*' as *yes*. The remaining discriminants were automatically marked as not belonging to the available parse trees. These discriminants are always stored after the one marked as *yes*.

```
704@7@4@6@named(Rei) GENDER masculine@@0@0@28-apr-2009 10:58
704@7@2@6@named(Rei) NUMBER singular@@0@0@28-apr-2009 10:58
704@7@4@6@named(Rei) PERSON 2nd-distant_or_3rd@@0@0@28-apr-2009 10:58
704@7@4@6@proper_q ARG0 named@@0@0@28-apr-2009 10:58
704@7@4@6@_interceder_v_1-afavorde-por- ARG1 named(Rei)@@0@0@28-apr-2009 10:58
```

Figure 5.9: Excerpt of the [incr tsdb()] `decision` file for the case in which the annotator marked the '*named(Rei) NUMBER singular*' as *no*. The remaining discriminants were automatically marked as not belonging to the available parse trees, but maintain their relative position to the one marked as *no*.

Annotator selects a discriminant as belonging to the correct parse tree

When an annotator decides that a discriminant does belong to the correct parse tree and marks it as *yes*, this discriminant is marked with a `1` value in the `decision` log files.

When one annotator marks a discriminant, at least one parse tree is discarded. All the discriminant which belong to the discarded parse trees are stored after the discriminant marked as *yes* with the value `4`. This is illustrated in Figure 5.8

Annotator selects a discriminant as not belonging to the correct parse tree

When a discriminant is marked as *no* by an annotator because it is considered not to contribute to the correct parse tree, this discriminant is marked with a `2` value in the `decision` log files.

The decisions concerning a discriminant will result in at least one parse tree being discarded. The discriminants belonging to the discarded parse trees are marked with a `4` and are stored in their original order relative to the discriminant marked as *no*. This is illustrated in Figure 5.9.

It is important to note that the discriminants in the *decision* log file are stored in

a default relative order. This order is the same order the discriminants are presented to the annotators in the annotation environment.

5.2.2 The `parse` log file

The parse log file contains information concerning the parsing of the sentence, such as the number of representations obtained and the time spent by the process.

```
8605@1@8605@2@10@10@10@0@-1@6@-1@-1@4564@163@109@34@98@0@2@223@240@-1@-1@2513564
@-1@-1@-1@22-03-2009 (13:10:10)@@ (:nmeanings . 0) (:clashes . 54) (:pruned . 0)
```

Figure 5.10: Content of the [incr tsdb()] log file `parse` for the sentence "Foi pura coincidência." ("It was pure coincidence.").

Much like the case of the `decision` log, the `parse` log contains different kinds of information related to the parsing process, also separated by characters `@`. Since each sentence only has one entry on the log file, we can take the one on Figure 5.10 to exemplify the different information fields:

- sentence identification, in this case '`8605`'.
- number of times the sentence was parsed, in this case '`1`'.
- sentence numeric identification (in our case is the same as the sentence identification) which is '`8605`'.
- number of grammatical representations obtained in this case '`2`'.
- time to achieve the first representation in milliseconds, being '`10`' for the example.
- time to achieve all representations in milliseconds which is also '`10`'.
- the CPU time spent parsing in milliseconds once more '`10`'.
- the garbage collection time used which is '`0`'.
- lexical rules retrieved '`6`'.
- number of tasks filtered as accepted by the parser '`4564`'.
- number of tasks executed by the parser '`163`'.
- number of succeeding tasks executed by the parser '`109`'.
- active items in the chart (edges) '`34`'.

```
8605@0@-1@-1@-1@-1@-1@-1@-1@-1@(129 root 2.71001e+17 ... (9 coincidência_1_noun
0 2 3 ("coincidência." 2 3))))))))))@@@ [ LTOP: h1 INDEX: e2 [ ... ] RELS: < ...
 > HCONS: < h1 qeq h9 h4 qeq h7 > ] @
```

Figure 5.11: Content of the [incr tsdb()] log file `result` for the sentence "Foi pura coincidência." ("It was pure coincidence.") with shortened derivation tree and MRS.

- passive items in the chart '98'.
- active items that contribute to the result '0'.
- passive items that contribute to the result '2'.
- number of unifications '223'.
- number of copy operations '240'.
- bytes of memory allocated '2513564'.
- date and time of parse '22-03-2009 (13:10:10)'.
- application specific comments '(:nmeanings . 0) (:clashes . 54) (:pruned . 0)'.

The fields left unmentioned are not used in the annotation or parsing process of the LXDeepBank, hence are left empty or have no relevant information.

5.2.3 The `result` log file

The `result` log file contains a list of all the possible grammatical representations for each sentence of the suite. The available parses are stored in the derivation tree format as well as the corresponding MRS. A full entry of the `result` log file is listed in Annex A.

The `result` log file stores the resulting possible representations from the parse process, containing the full derivation tree and MRS. The example in Figure 5.11 had both this representations severely shorten since those representations are not the main focus of this document. For the sake of completeness, the full representation is available in Annex A. The fields on this representation are as follows:

- sentence identification, the name or number of the sentence, in this case '8605'.
- result integer; 0 if no error is detected.
- the derivation tree, which contains the grammatical rules used as well as their span and statistical information, starts with the rule root '(129 root' and was shortened for presentation purposes.

- the MRS representation, which is also fairly shortened in Figure 5.11 since the information contained in it is not relevant for this subject, starts with the '`[ LTOP: h1`' and ends with the lists of arguments '`< h1 qeq h9 h4 qeq h7 > ]`'.

The fields left unmentioned are not used in the annotation or parsing process of the LXDeepBank, hence are left empty or have no relevant information.

5.2.4 The `preference` log file

The `preference` log file is the simplest of all the [incr tsdb()] log files used. It contains the number of the grammatical representation considered the correct by the annotator and the number of the revision in which it was selected.

```
8605@4@0
```

Figure 5.12: Content of the [incr tsdb()] log file `preference` for the sentence "Foi pura coincidência." ("It was pure coincidence.").

The preference log file (Figure 5.12) is a very simple and concise list of triples with the sentence identification, the revision number and the number of the grammatical representation selected by the annotator.

It is important to point out that only the sentences considered to have a correct grammatical representation are present. However, if a grammatical representation is accepted for a sentence and afterwards that sentence is rejected, the first attribution can still be present.

The fields are once more separated by @'s and taking the entry on Figure 5.12 as a reference, the fields are:

- sentence identification. (the name or number of the sentence); in this case '`8605`'.
- the number of the revision on which this decision was made; in this case it was revision `4`.
- the identification of the grammatical representation found to be the most correct one for sentence `8605`; in this case, it was the representation `0`, which is the first grammatical representation found for the sentence.

The fields left unmentioned are not used in the annotation or parsing process of the LXDeepBank, hence are left empty or have no relevant information.

5.2.5 Summary

The LOGON tool collection was designed for grammar engineers, and both user interface and functionalities were optimized with this purpose in mind. It can also be used for corpora annotation thanks to the support for the used grammar (LXGram in our case) and the graphic user interface that allows the annotators to see for instance a constituency tree representation of the possible grammatical representations of the sentence being computed while the annotation is performed.

5.3 Decision comparison

In order to measure the agreement between annotators, it is necessary first to recover the annotators decisions for each sentence, and second to compare these decisions between annotators.

The annotators make a certain number of decisions when annotating a sentence, and the [incr tsdb()] aids the annotator by automatically classifying all the discriminants that can be classified as an automatic consequence of any annotator manual decision. For example, in case the annotator decides a given discriminant belongs to the correct grammatical representation for the sentence being considered, the [incr tsdb()] automatically classifies all the discriminants not belonging to the representation that contains the manually assigned discriminant as such. To the set containing the manual decision and all the automatically associated decisions we term it an option.

The amount of data available, the way it is stored by the [incr tsdb()], and the not always direct correspondence between the order of decisions by different annotators turn what seems at first blush a trivial task into a somewhat complex task.

As it was described in the previous Chapter 5.2.1, there are different ways to reach the same final result in the `decision` log file. The need to unify these heterogeneous values requires a non trivial approach.

The approach used to solve this particular problem consists in grouping the decisions in the decision log file according to their type. Since type `1` decisions are always placed before the automatic decisions resulting from it, both the decision of type `1` and the following automatic decisions can be grouped together. The decisions of type `2` however appear in the original position of the discriminants. Hence, both the previous and following discriminants need to be grouped with it in order to assert these decisions values.

These groupings help to better compare different sets of discriminants that result in identical parses being selected by both annotators.

5.3.1 Discriminant comparison

The discriminant comparison consists in comparing the discriminants of both annotators side-by-side using the algorithm represented in Figure 5.13. The input are the decision logs of annotator A and annotator B. Lists A and B are scanned entry by entry.

The Algorithm

The algorithm has as input the content of the *decision log file* for each annotator in the format of a list of lines, one line per entry.

Since one of the decision lists of a such annotator can have more entries than the list of the other annotators (for example because one of the annotators rejected the sentence after considering some discriminants), the algorithm starts by setting the largest list as the reference list, that is, the one on the outer search cycle. This helps solve problems such as one list ending before the other.

The algorithm searches for the next manual decision (type 1 and 2) on both lists in a zig-zag scan fashion. When one manual decision is found, the neighborhood of this decision for both annotators is delimited to be taken into consideration. This neighborhood is the set of decisions relative to the same discriminant the manual decision refers to. When considering neighborhoods three independent cases have to be considered:

Rejection sets:

Taking as reference the content of Figure 5.14, when the entry "`...@2@_claro_a_1 ARG1 _relação_n_2-de-por-@...`" is analyzed, the neighborhood of this decision is delimited. This consists in searching from the start of the set of decisions with the same label in the decision list for decisions with the same discriminant label as the manual one, which in this example are all the discriminants with the discriminant label "`_claro_a_1`". The neighborhood taken into account is represented in Figure 5.15. The decisions of each annotator are compared using the discriminant label as reference. Since decisions of type `4` result from discriminants automatically rejected due to adjacent decisions by annotator A, it can be concluded that a similar set of options of 4's and 2's by Annotator B can be considered equivalent even if different discriminants were manually selected, since the end result was the same rejection set.

Hence the set of decisions in Figure 5.15 are considered similar and the counter of the number of equal options is incremented by one.

Acceptance sets:

Since the manual acceptance decisions (type `1`) are placed on the top of all au-

input: Decision List A, Decision List B
data format: Lists with a decision per entry
data initialization: $countTotal := 0$ $countIdentical := 0$

1. While there are options left in any of the decision lists:
   - 1.1. Find next manual option d_X
     - 1.1.1. Zig-Zag through both lists, stopping at the first manual option (option type 1 or 2) from the list of annotator X.
   - 1.2. Determine neighborhood for d_X
     - 1.2.1. Neighborhood for d_X in X, N_X^d
       if decision is type 1: $N_X^d := d_X$ and the discriminants below with the same discriminant label
       else if decision is type 2: $N_X^d := d_X$ and the discriminants above and below with the same discriminant label
     - 1.2.2. Mirror Neighborhood of d_X in $\bar{X}$, $N_{\bar{X}}^d$
       $N_X^d :=$ lines with the same discriminant label as d_X
     - 1.2.3. Else (Stranded neighborhood)
       $N_X^d := empty$
   - 1.3. Compare options
     - 1.3.1. flag := true
     - 1.3.2. *if* N_X^d *is empty or* $N_{\hat{X}}^d$ *is empty:* $flag := false$
     - 1.3.3. while flag is true
       - 1.3.3.1. for each line in N_X^d
         for each line in $N_{\hat{X}}^d$
         if discriminant description match and not equivalent type of option
         $flag := false$
   - 1.4. Count equivalent options
     - 1.4.1. $countTotal := countTotal + 1$
     - 1.4.2. if(flag)
       $countIdentical := countIdentical + 1$
   - 1.5. Cleanup
     - 1.5.1. remove neighborhood from list A and B
2. Return $countTotal - countIdentical$

Figure 5.13: The pseudo-code of the decision comparison algorithm.

tomatically made decisions related to it when the decision is "`...@1@_de_p ARG1 _claro_a_1@...`", it is only necessary to take into account this decision and the ones (of type 2, 3 and 4) that follow it until the next human decision for a different discriminant is found. That removes the necessity to look for neighborhood members previous to this decision.

Figure 5.16 represents the considered decisions, the decisions of type `1` and all following decisions of type `3` or `4` with the same discriminant label are considered. If all have equivalent values, the counter of the number of equal options is incremented

Annotator A's decision log file	**Annotator B's decision log file**
...@**4**@_claro_a_1 ARG1 _relação_n_1-com-de-@...	...@**4**@_claro_a_1 ARG1 _relação_n_1-com-de-@...
...@**2**@_claro_a_1 ARG1 _relação_n_2-de-por-@...	...@**4**@_claro_a_1 ARG1 _relação_n_2-de-por-@...
...@**4**@_claro_a_1 E.ASPECT.PERF -@...	...@**2**@_claro_a_1 E.ASPECT.PERF -@...
...@**4**@_claro_a_1 E.MOOD indicativo@...	...@**4**@_claro_a_1 E.MOOD indicativo@...
...@**1**@_de_p ARG1 _claro_a_1@...	...@**1**@_de_p ARG1 _claro_a_1@...
...@**4**@_de_p ARG1 _claro_a_2@...	...@**4**@_de_p ARG1 _claro_a_2@...
...@**2**@_em_p ARG1 _relação_n_1-com-de-@...	...@**1**@_em_p ARG1 _relação_n_1-com-de-@...
...@**4**@_em_p ARG1 _relação_n_2-de-por-@...	...@**4**@_em_p ARG1 _relação_n_2-de-por-@...
...@**4**@_muito_a ARG1 _claro_a_1@...	...@**4@**6@_muito_a ARG1 _claro_a_1@...
...@**2**@_muito_a ARG1 _claro_a_2@...	...@**4**@_muito_a ARG1 _claro_a_2@...
...@**4**@_muito_x ARG1 _claro_a_1@...	...@**4**@_muito_x ARG1 _claro_a_1@...
...@**1**@_muito_x ARG1 _claro_a_2@...	...@**2**@_muito_x ARG2 _claro_a_2@...

Figure 5.14: Two annotators `decision` log file content side-by-side comparison for a specific sentence. With some fields removed for easier interpretation.

Annotator A's decision log file	**Annotator B's decision log file**
...@**4**@_claro_a_1 ARG1 _relação_n_1-com-de-@...	...@**4**@_claro_a_1 ARG1 _relação_n_1-com-de-@...
...@**2**@_claro_a_1 ARG1 _relação_n_2-de-por-@...	...@**4**@_claro_a_1 ARG1 _relação_n_2-de-por-@...
...@**4**@_claro_a_1 E.ASPECT.PERF -@...	...@**2**@_claro_a_1 E.ASPECT.PERF -@...
...@**4**@_claro_a_1 E.MOOD indicativo@...	...@**4**@_claro_a_1 E.MOOD indicativo@...
⋮	⋮

Figure 5.15: Neighborhood of the manually rejected decision " `...@2@_claro_a_1 ARG1 _relação_n_2-de-por-@...`"

by one, otherwise only the number of total options is incremented.

Annotator A's decision log file	**Annotator B's decision log file**
...@**1**@_de_p ARG1 _claro_a_1@...	...@**1**@_de_p ARG1 _claro_a_1@...
...@**4**@_de_p ARG1 _claro_a_2@...	...@**4**@_de_p ARG1 _claro_a_2@...
⋮	⋮

Figure 5.16: Neighborhood of the manually accepted decision " `...@1@_de_p ARG1 _claro_a_1@...`"

Special sets:

In some cases, the discriminant lists for the annotators can have a different number of decisions and still support the same grammatical representation of the sentence being annotated.

For instance, in some cases it is possible for the acceptance of one discriminant by annotator A and the rejection of a different discriminant by annotator B to result in the same grammatical representation. That is why if a neighborhood in list A starts with a manual decision of 1 and the other does not match that initial decision, but the set of automatic decisions of annotator A is identical to the set of decisions of annotator B, that option considered to be identical. In the example of Figure 5.17, annotator A has 2 decisions for the discriminant "_de_p" and annotator B only has one, however, once neighborhoods are defined and the decisions are compared, the outcome is considered to be equivalent.

That is why the algorithm cross-compares the decisions in the neighborhood of a manual decision of annotator A with the decision in the neighborhood of a manual decision of annotator B.

Annotator A's decision log file	**Annotator B's decision log file**
...@**1**@_de_p ARG1 _claro_a_1@...	...@**2**@_de_p ARG2 _claro_a_1@...
...@**4**@_de_p ARG2 _claro_a_1@...	...@**1**@_em_p ARG1 _relação_n_1-com-de-@...
...@**2**@_em_p ARG1 _relação_n_1-com-de-@...	
⋮	⋮

Figure 5.17: Portion of the decision log considered when method is called for the entry " `...@1@_de_p ARG1 _claro_a_1@...`"

There is also other possible special case. In this case, there are manual decisions on one of the annotators decision lists which were not considered at all by the second annotator, which means they are not present at all in the second annotators list.

This kind of cases are identified when a neighborhood for a specific discriminant label can not be found in the other annotators list. In this case, all manual decisions are considered for the total number of decisions, but none is counted as identical since there is no similar decisions for the other annotator. Figure 5.18 represents one of these cases, where the decision "**1**@_de_p ARG1 _claro_a_1@..." is unilateral to annotator A.

Annotator A's decision log file	**Annotator B's decision log file**
...@**1**@_de_p ARG1 _claro_a_1@...	...@**1**@_em_p ARG1 _relação_n_1-com-de-@...
...@**1**@_em_p ARG1 _relação_n_1-com-de-@...	
⋮	⋮

Figure 5.18: Portion of the decision log wth a unilateral stranded decision.

This information is then used to calculate levels of agreement between the two annotators comparing the decisions both took either similar or divergent.

5.4 Coefficient Adaptation

Some adaptations were necessary in order to make an optimal usage of the data available in the [incr tsdb()] log files. Most adaptations are related to the analysis and interpretation of the available data and the format it is available in.

5.4.1 Issues with [incr tsdb()] log files

The main issues related to the analysis and utilization of the [incr tsdb()] logs are the following:

- Not all possible discriminants are stored in the log files. Only the decisions made directly by the human annotator and the direct consequences of those

decisions are stored in the [incr tsdb()] log files. This issue can be illustrated by the examples of Section 5.2.1.

- An annotator can reject a sentence at any given moment. If no decisions were made on any discriminants, no information about discriminants is stored. In case of rejection, only the decisions taken until the rejection are stored in the log files, which in practice can be none. This problem is illustrated in Figure 5.19, where the annotator rejected the sentence after marking the discriminants '*cardinal ARG2 greater-or-equal*' and '*_cerca+de_x ARG int-equals(200)*' as *yes*.

- If a sentence only has one possible parse and therefore requires no decisions regarding the discriminants, no information about the available discriminants or the implied decisions to achieve the desired parse are stored in the log files. The Figure 5.20 shows what the annotation interface looks like when only one parse was generated for the sentence.

  This results in a empty `decision` log file and a simple '`207@1@0`' entry in the `preference` log file.

- Due to some internal bugs or lack of resources, sometimes the [incr tsdb()] crashes while processing sentences and these sentences are not recovered during the annotation. If this happens, the sentence will have no reference whatsoever in one of the annotators log files. These sentences are not considered for agreement calculation purposes and are known as the *lost sentences*.

```
303@1@-1@5@@@0@11@27-jan-2011 10:36
303@1@1@6@cardinal ARG2 greater-or-equal@@0@0@27-jan-2011 10:35
303@1@4@6@times FACTOR1 plus@@0@0@27-jan-2011 10:35
303@1@1@6@_cerca+de_x ARG1 int-equals(200)@@0@0@27-jan-2011 10:36
303@1@4@6@_cerca+de_x ARG1 plus@@0@0@27-jan-2011 10:35
303@1@4@6@_cerca+de_x ARG1 times@@0@0@27-jan-2011 10:35
303@1@4@6@_comprar_v_-a- ARG2 _computador_n@@0@0@27-jan-2011 10:35
```

Figure 5.19: Content of the [incr tsdb()] log file `decision` for the case where the annotator rejects a sentence after deciding on some of the available discriminants.

5.4.2 Approximation to the theoretic agreement coefficient model

The only available information regarding the annotation process is the information contained in the [incr tsdb()] log files. As it was detailed in the previous section, this information is not exhaustive and some gaps may exist.

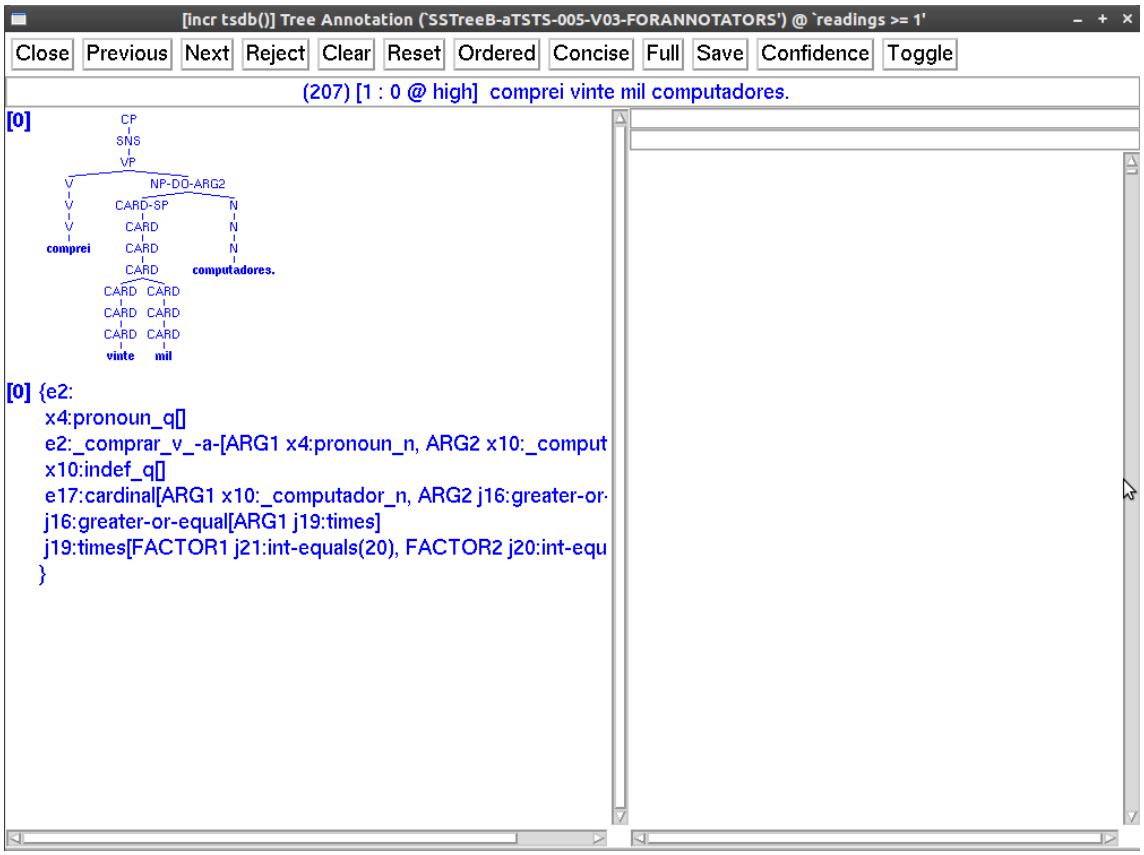


Figure 5.20: The [incr tsdb()] annotation interface when annotating a sentence with a single parse. Where only parse is represented both by the constituency tree and logical form.

Different courses of action in the annotation process can result in different sets of discriminants being stored. With the intent of minimizing the impact of these gaps in the overall result, the sentences of the corpus are divided in three subsets.

Considering S the set of sentences with at least one available parse, these sentences are divided into the following subsets:

1. ***Sentences accepted by both annotators*** c_1 ***and*** c_2***.***

   For the sentences for which each annotator selected a grammatical representation, the logs allow to obtain all the options necessary to generate these representations. These are the options considered as O_s, the options for sentence s.

   This subset of corpus S can be called S_{both}, and $S_{both} \subseteq S$. The observed agreement (A_o^s) for this sentences can be calculated using formula 4.1 (see page 42), and the expected agreement (A_e^s) can likewise be considered the uniform distribution of binary decisions, which is 0.5.

   It is practical to gather some extra information regarding the sentences for

which the annotators disagreed since these sentences have the maximum amount of information that is possible to recover from these log files.

With this in mind, the proportion of divergence (P_D)is calculated. The proportion of divergence for sentence s can be calculated as the ratio of different options and total options available. For this we need the notion of divergent options for sentence s, which is the total number of options for s O_s minus the options equal to both annotators (O_s^{eq}).

$$O_s^{div} = O_s - O_s^{eq} \tag{5.1}$$

$$P_D^s = \frac{O_s^{div}}{O_s} \tag{5.2}$$

This notion can be generalized for the whole sub-corpus S_{both} by calculating the ratio between the sum of the divergent decisions for all $s \in S_{both}$.

$$P_D^{S_{both}} = \frac{\sum_{s \in S_{both}} O_s^{div}}{\sum_{s \in S_{both}} O_s} \tag{5.3}$$

An example can help to understand this notion, taking Table 5.1. With this information, it is easy to calculate:

$$P_D^{S_{both}} = \frac{4+1+9+3+5}{12+7+9+10+17} = \frac{22}{55} = 0,40 \tag{5.4}$$

$\mathbf{\lvert O_s \rvert}$	$\mathbf{O_s^{eq}}$	$\mathbf{O_s^{div}}$	$\mathbf{P_D^s}$
12	8	4	0,33
7	6	1	0,14
9	0	9	1,00
10	7	3	0,30
17	12	5	0,29

Table 5.1: Sample of option values for 5 sentences with divergences.

This information can be used to reduce the impact of the information gaps left in the [incr tsdb()] log files, since in cases where some information was lost this proportion can be used to estimate the annotators divergence.

2. ***Sentences rejected by at least one annotator***

As mentioned before in Section 5.4.1, in case of rejection, the only options available on the [incr tsdb()] log files are the ones made to the point of rejection, which can result in cases where sentences may have different sets of options

for each annotator. Due to this design of the log file system, the options of a sentence rejected by at least one annotator have to be divided into two subsets: The options common to both annotators Os^{com} and the options only present for one annotator O_s^{uniq}.

The options taken by both annotators, can be compared in the same way as the ones present in sentences accepted by both annotators.

However, regarding the options taken only by one annotator there is a lack of information about such options by the other annotator. Therefore an estimation needs to be made. Since the proportion of disagreement $P_D^{S_{both}}$ is available, it is possible to calculate in which proportion the options unique to one annotator log would be divergent if considered by both annotators. That is, for a set of options unique to one of the annotators for sentence s or O_s^{uniq}, it is possible to obtain an estimation for the number of options on which the annotators would agreed if both had considered them. This can be called $\widehat{O_s^{eq}}$ and is calculated using the following equation:

$$\widehat{O_s^{eq}} = |O_s^{uniq}| \cdot (1 - P_D^{S_{both}}) \tag{5.5}$$

With this information, it is possible to calculate the observed agreement for any sentence $s \in S_{R1}$ (sentences rejected by at least one annotator):

$$A_o^s = \frac{O_s^{eq} + \widehat{O_s^{eq}}}{O_s} \tag{5.6}$$

Where O_s^{eq} are the options for sentence s on which both annotators fully agree and $\widehat{O_s^{eq}}$ the estimated number of options only considered by one annotator for which both annotators would have agreed upon.

Table 5.2 as a sample of five sentences where some of the options were taken only by one of the annotators and hence their agreement had to be estimated. The agreement without estimation is also present in the table. This means that for this agreement calculation the options considered only by one annotator are considered as completely divergent.

Using the information gathered from the sentences accepted by both annotators (S_{both}) and from the sentences rejected by at least one annotator (S_{R1}), it is possible to approximate the agreement coefficient for the corpora O_S.

3. ***Sentences without stored options***

   As pointed out previously in 5.4.1, if there is only one possible grammatical representation, or if an annotator rejects a sentence without taking any options, the [incr tsdb()] stores no information about the available or implied

$\mathbf{O_s}$	$\mathbf{O_s^{com}}$	$\mathbf{O_s^{eq}}$	$\mathbf{O_s^{uniq}}$	$\widehat{\mathbf{O_s^{eq}}}$	**With estimation** $\mathbf{A_o^s}$	**Without estimation** $\mathbf{A_o^s}$
14	10	8	4	2	$0,71$	$0,57$
8	2	1	6	3	$0,50$	$0,13$
6	4	3	2	1	$0,67$	$0,50$
12	8	3	4	2	$0,42$	$0,25$
9	5	4	4	2	$0,67$	$0,44$
Average Value					$0,59$	0.38

Table 5.2: Agreement for 5 sentences over the options taken only by one of the annotators. In this table, the value of $P_D^{S_{both}}$ is used in token (5.4).

options. These cases are what the third subset of sentences consists on, the sentences without any stored options or S_{noop}.

Since no option information is available for these options, it is necessary to estimate an average of options per sentence for the sentences of the corpora. This can be done by calculating the ratio of options per sentence for the two previous subsets of sentences, both the sentences accepted by both annotators S_{both} and the sentences rejected by at least one annotator S_{R1}. This estimate can be calculated by the following equation:

$$\widehat{O_{avg}} = \frac{|O_{S_{both}}| + |O_{S_{R1}}|}{|S_{both}| + |S_{R1}|} \tag{5.7}$$

For example, if the two sets of sentences accepted by both annotators and of the sentences rejected by at least one have respectively 20000 and 6000 considered options and 1400 and 600 sentences respectively, the number of average options per sentence is:

$$\widehat{O_{avg}} = \frac{26000}{2000} = 13 \tag{5.8}$$

With this value, it is possible to calculate an estimation of the observed agreement for sentences with no recorded options by calculating the proportion of disagreement of the average decisions on which the annotators would have agreed upon if the tool has stored their options. This can be obtained for any sentence $s \in S_{noop}$ using this equation:

$$A_o^s = \frac{\widehat{O_{avg}} - (\widehat{O_{avg}} \cdot P_D^{S_{both}})}{\widehat{O_{avg}}} \tag{5.9}$$

Once these values are calculated, it is only a matter of calculating the K_Y^s for each sentence of the corpus S using formula 4.2 from page 43. The agreement for the entire corpus S can be likewise be calculated using formula 4.3 from page 43.

5.4.3 Summary

With the objective of minimizing the impact of the information gaps in the [incr tsdb()] log files, some estimations are made. These estimates are an educated guess of what the values would be if they could be accessible.

5.5 Testing and results

5.5.1 Test Corpus

The parsed corpus used for this experiment consisted on 12319 sentences to be evaluated from which 3297 were accepted and 6684 were rejected by both annotators, the rest of the sentences were accepted by one of the annotators.

5.5.2 Results

When assessing an evaluation metric or tool it is necessary to have a term of comparison to better gauge the value of the results obtained.

In order to apply the inter-annotator agreement metric presented in this document, an application was developed. The details of this application are further explained in the next chapter.

With this application and using the LXDeepGramBank annotation logs, the agreement values were calculated for the K_Y agreement coefficient.

The data used for this test was the version 3 of the LXDeepGramBank and the result is better than the one expected. The agreement score obtained was of Y-Option Kappa = 0.86 which is over the threshold of the 0.80 widely used in literature. This score can be considered very positive since all sentences were taken into account and all options made by the annotators were considered.

As an exercise, the annotated corpora was manually exported to the treebank format. The grammatical representations picked by the annotators were then compared using the Parseval metric by means of the F-Score, as suggested by Brants (2000). The Leaf-Ancestor metric was also calculated since it is being used more frequently to evaluate the same kind of tasks as Parseval. As mentioned before, both these metrics are usually used for parser output evaluation.

For the inter-annotation coefficients usually the interpretation of one annotator is considered to be the gold standard to which the interpretation of the other annotator is compared to. However, in this task not all sentences have a valid interpretation

or a interpretation at all by both the annotators[2], which leaves some sentences with a unilateral interpretation.

Due to this problem, only 56.80% of the sentences were considered (the ones with a tree representation for both annotators). Using the sentence comparison, the F-Score of 98% was obtained. If the sentences that have a tree representation by only one of the annotators are considered, however, considering that these sentences have an F-score of 0% since there is no comparison grounds, the total F-Score drops to 55%.

The Leaf-Ancestor metric yields very similar results. The set of sentences that could be compared side by side obtained a Leaf-Ancestor score of 96.60%, but when all the sentences are considered the Leaf-Ancestor fails to yields any results for empty trees since the tool used to calculate the metric does not support empty trees as input.

As to the EDM metric, unfortunately it was not possible to calculate it since it requires a characterization to be applied meaningfully (the span information for each token present on the example on Chapter 3.2.2) which is not supported by the current NLX pipeline (the preprocessing and PET parser parts). Introducing such information would require an adaptation of this pipeline and the subsequent re-parsing and re-annotation of the entire corpora, which would require several person-month if not person-year of additional work.

It is important to note that, on the one hand, a Kappa coefficient and, on the other hand, the Parseval or F-Score scores are not comparable and this exercise was only conducted to assess the values obtained by the available corpora.

5.5.3 Summary

The F-Score improves the reliability of the Parseval metric. However it is still very coarse-grained and sentences that do not yield a constituency tree result in an F-Score of 0. The same can be said for the Leaf-Ancestor metric.

The Y-Option Kappa however is much more granular when it comes to levels of agreement, which is reflected in terms of percentage agreement or observed agreement. And the increase in granularity allows for a much more accurate assessment when it comes to text produced as the outcome annotation tasks.

The estimates that result from lack of available information can in the future be solved by either a patch to some of the tools in the LOGON collection or the development of an dynamic annotation oriented tool that stores all the information for later use.

[2]In some cases only one of the annotators selected a grammatical representation as a valid representation for the sentence.

Chapter 6

Implementation

The agreement metric is useful only if it can be applied to practical cases. With that ourpose in mind, a tool was developed that deals with the [incr tsdb()] log files and computes the necessary data and a spreadsheet report with the most relevant data and results.

Since the main goal of this application is to compare annotations made using the [incr tsdb()] tool over the LKB development environment, the tool is called CompAnnotLkb. It is fully implemented in Java[1] with the aid of the Eclipse programming environment[2].

6.1 Application Structure

The application operation can be divided into four major parts.

- Gathering of information: The [incr tsdb()] log files are read into memory and the relevant data is extracted and stored into appropriate formats.
- Data analysis and validation: The stored data is analyzed and filtered; for example, the lost sentences are identified and discarded. Some statistical data is gathered and stored, such as the number of sentences per corpus, how many had at least one valid parse, and so on.
- Agreement Metric Computation: By using the gathered and filtered information, the agreement metrics are computed.
- Output formatting and result presentation: A spreadsheet format report is generated with all the relevant data extracted from the [incr tsdb()] log files and calculated metrics.

[1] Java is a popular object oriented language developed by Sun Microsystems now part of Oracle and is available at: http://www.java.com/

[2] Eclipse SDK is available at: http://www.eclipse.org/

Following the Java framework, the application is divided into packages, each associated with different functions and roles. A class diagram of the public members of the applcation can be seen in Annex B. The existing packages are the following:

pt.ul.fc.di.nlx.compAnnotLkb.main contains the test executable class that allows user interaction by mean of several options related with the execution and configuration of then application execution.

pt.ul.fc.di.nlx.compAnnotLkb.holders as indicated by the name, contains all the structures that hold data, from the content of files to specific data for metric computation.

pt.ul.fc.di.nlx.compAnnotLkb.io.input contains all classes related to the acquisition of information, including opening and reading of files.

pt.ul.fc.di.nlx.compAnnotLkb.util.svn contains the svn implementation and adaptations that allow to fetch and store data in and from a svn repository if such is specified.

pt.ul.fc.di.nlx.compAnnotLkb.util contains the general utility class that contains small methods or operations that are repeated throughout the application implementation but do not quite belong to any specific class.

pt.ul.fc.di.nlx.compAnnotLkb.coefficients contains the classes which implement the computation of the desired agreement coefficients.

pt.ul.fc.di.nlx.compAnnotLkb.io.output contains the classes related to the output of data, specifically the writing of the spreadsheet data and result report.

6.2 Application Operation

To gather all the available information about the annotation, it is necessary to first find the information, and secondly to identify the information which should be considered.

6.2.1 Configuration files

To aid this process, the compAnnotLkb has a configuration file (also available via execution arguments) where the necessary information can be set. This configuration information assumes the input data obeys the same organization as in Figure 6.1 and the most important fields are:

TEMP_PATH a working directory where the working data is located and the output files will be stored during execution. In case SVN is used, a working copy is extracted to this location and can be analyzed locally. Otherwise, the data should be already in this location.

PATTERN the pattern to look for in the suite directories to make sure only valid ones are searched. This is necessary since in some cases extra data is available in the suites directories such as svn state directories and other directories waiting for annotation which usually have a different name from the already annotated ones. With this pattern, the application only has to consider valid directories saving some execution time.

This pattern can be a regular expression, for example the pattern "$^SUITE-[0-9]+-V[0-9]+\$$' indicates that only directories that start with SUITE have a - and than a a serie of numerals curresponding to the suite number and finally a - and a V followed by numerals indicating the version, are considered valid for evaluation.

DIR1/2 the directories to analyze in the *temp_path*; since there may be more than 2 annotators and data is analyzed for pairs of annotators, it's important to tell the application which annotators to consider; this is easily done if the appropriate annotators directories are indicated.

6.2.2 Data Structures

The application starts by identifying equivalent suite directories for both annotators. Since some operations are necessary over the content of the log files, their content is loaded into memory on the format of a list of text lines.

These files content is filtered and the relevant data is stored in specific data structures that can be easily searched and listed in future tasks. Before these structures are described, it is necessary to describe some minor data structures used to hold the necessary information for one sentence.

Basic Data Structures

Sentence this data structure contains the necessary information for a given sentence. It has three fields: the number of the sentence the structure refers to; the number of the last revision for the sentence and finally if accepted the number of the accepted grammatical representation.

SentenceParseTree a data structure used to hold how many grammatical representations each sentence has in this version. It has two fields: the number of

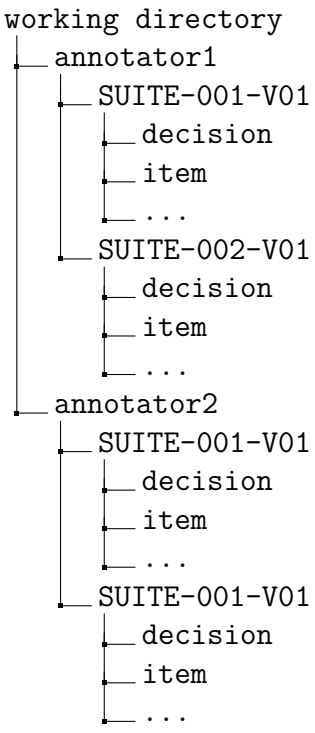


Figure 6.1: Example of a working directory for the compAnnotLkb application where annotator 1 and 2 analysis for the suites 1 and 2 are to be analyzed.

the sentence the structure refers to; the number of grammatical representations for the sentence.

SuitePair contains the data refering to a sentence, but with additional information about the suite and the annotator. It has 4 fields: the number of the sentence the structure refers to; the number of the suite the sentence is part of; additional annotation information, for example if only one annotator accepted this sentence which annotator was or which grammatical representations each annotator picked if both accepted but diverged; the corpus this suite is part of, for example the test corpus is the *aTSTS*.

Composite Data Structures

The composite data structures are lists or sets of information for each annotator or for the entire corpus, and most the basic data structures introduced previously.

acceptedA/B list of sentences accepted by the annotators. There is one for each annotator and contain a *Sentence* entry for each sentence.

rejectedA/B[3] list of sentences accepted by the annotators. There is one for each annotator and contain a *Sentence* entry for each sentence.

parseTrees list of parsed sentences in the *SentenceParseTree* format. It contains one entry for each sentence that has at least one valid grammatical representation.

decision list of decisions for each sentence. The entries are in the *TreeDecision* format.

TreeDecision this data structure is slightly more complex than the previous mentioned. It holds the necessary information about the decisions of each annotator. It has eight fields: the number of the sentence the structure refers to; list of pairs option / decision for each annotator per sentence; last revision for each annotator; total number options available for the sentence; set of options present on both annotators lists.

After this structures are properly loaded the divergence list is calculated for each suite of each corpus, this list contains entries of the type *SuitePair* and contains only information about sentences on which the annotators analysis diverged.

The execution of the application follows the flow depicted on Figure 6.2.

6.2.3 Computation of the metric

The information loaded into the aftermentioned structures is passed to the implementation of the agreement metrics and the necessary computations are made. In the case of the Y-Option agreement coefficient, the data is divided into the three sets described in Chapter 5.4.2 and each considered accordingly.

After the computation of the agreement metrics and related data takes place, a spreadsheet format report is generated with all he necessary data. This part consists of a spreadsheet with all the relevant information to the adjudicator. The front page of the report (Figure 6.3) contains the general information about when the analysis was made and the corpus analyzed as well as the metrics values and the list of divergences.

The *data* (Figure 6.4) sheet has all the relevant information the partial data about the accepted and rejected sentences as well as the one related with the coefficients as well as the sums used during their computation.

[3]The lists of accepted and rejected sentences are separated because for some tasks only one o the lists is relevant and therefore it is more efficient to separate the lists and work only with the needed one and only go through both when strictly necessary.

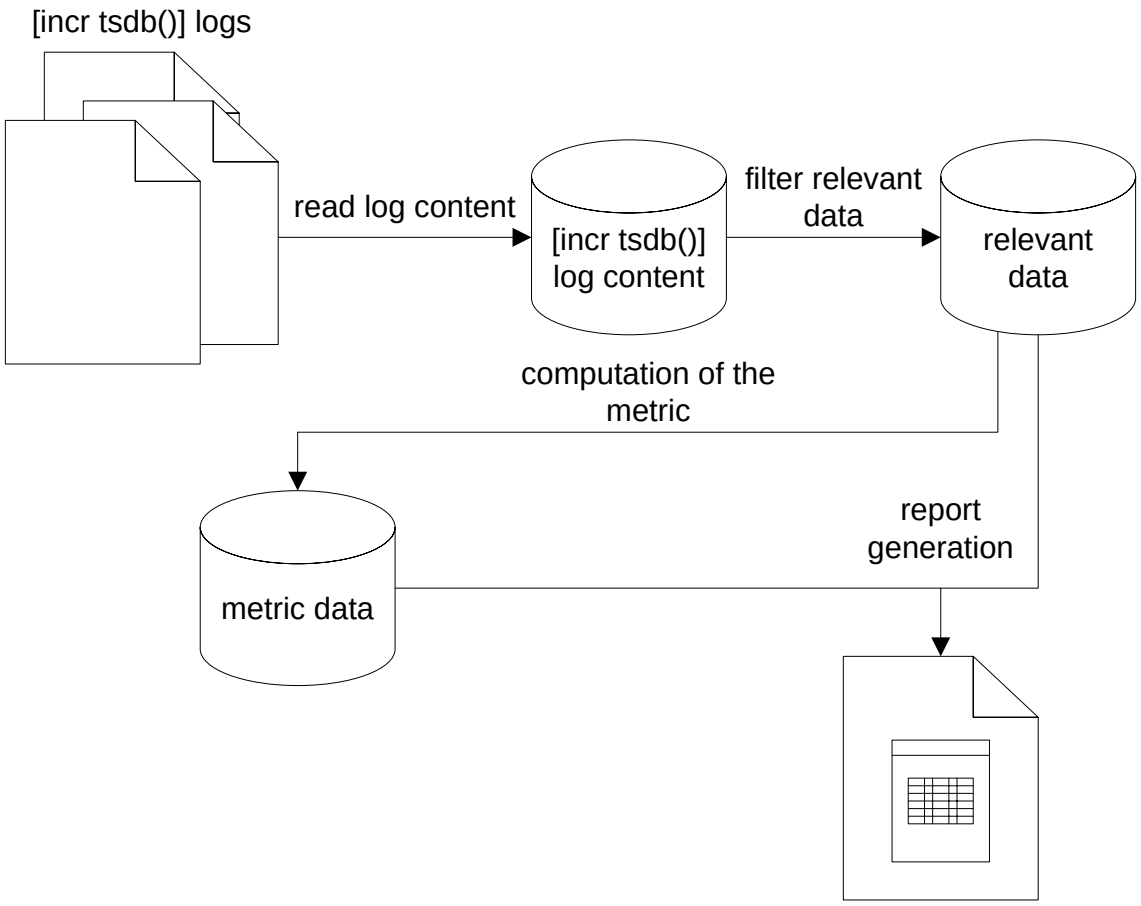
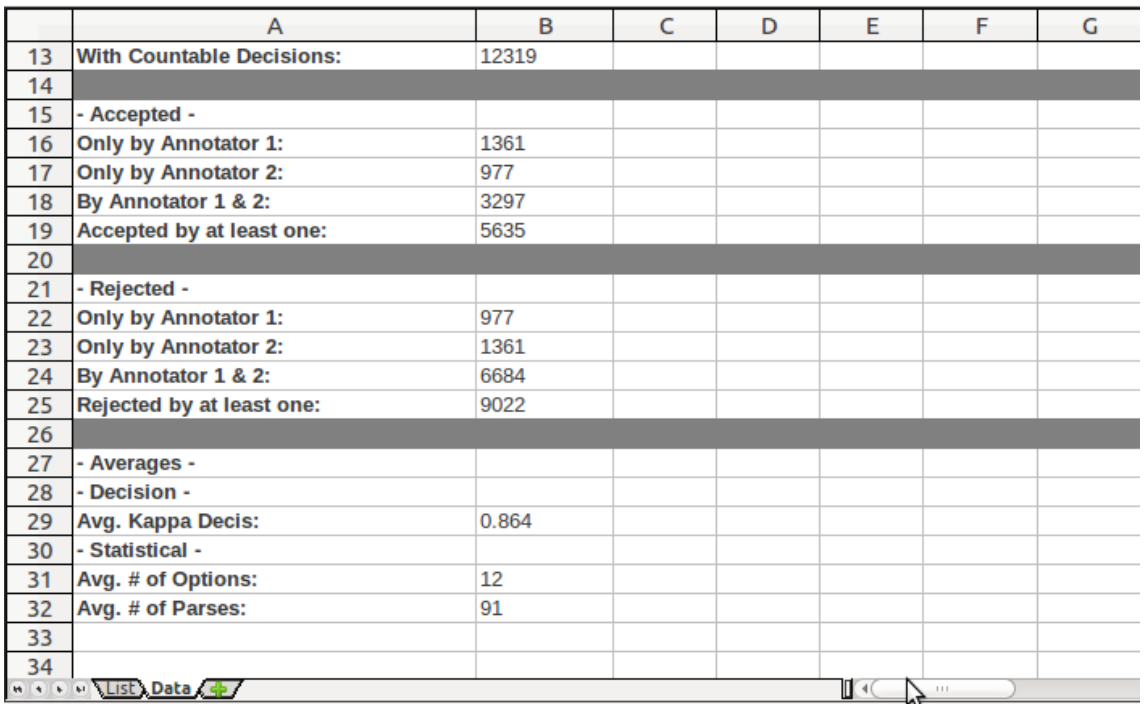


Figure 6.2: The CompAnnotLkb execution flow.

	A	B	C	D	E	F	G
1	**Type:**	**Comparison for annotators of Treebank**					
2	**Tool used:**	compAnnotLkb					
3	**Date:**	25 07 2011 13:49					
4							
5	**Parsed Sentences:**	15481					
6	**Suite Distribution:**						
7	**aTSTS**	Start Suite:	1	End Suite:	18	Sentences:	874
8	**bCINT**	Start Suite:	1	End Suite:	239	Sentences:	3335
9	**cCINT**	Start Suite:	1	End Suite:	52	Sentences:	775
10	**dPENN**	Start Suite:	1	End Suite:	16	Sentences:	279
11	**eCTMP**	Start Suite:	1	End Suite:	680	Sentences:	10218
12	**Sentences accepted by at least one:**	5635					
13	**Sentences rejected by at least one:**	9022					
14	**- K Coefficients -**						
15	**Avg. Y-Option Kappa:**	0.864					
16							
17	**Suite number**	**Sentence number**	**Accepted only by:**	**Comments:**			
18	aTSTS - 1	6	Annotator1 - 4				
19	aTSTS - 1	7	Annotator1 - 10				
20	aTSTS - 1	8	Both - 4/10				
21	aTSTS - 1	9	Both - 0/1				
22	aTSTS - 1	10	Annotator1 - 0				
23	aTSTS - 1	11	Both - 0/1				
24	aTSTS - 1	13	Annotator1 - 1				
25	aTSTS - 1	17	Annotator3 - 0				
26	aTSTS - 1	18	Annotator3 - 0				

List Data

Figure 6.3: Front sheet of the compAnnotLkb report with all the data relevant to the adjudication process of the annotation.



	A	B	C	D	E	F	G
13	With Countable Decisions:	12319					
14							
15	- Accepted -						
16	Only by Annotator 1:	1361					
17	Only by Annotator 2:	977					
18	By Annotator 1 & 2:	3297					
19	Accepted by at least one:	5635					
20							
21	- Rejected -						
22	Only by Annotator 1:	977					
23	Only by Annotator 2:	1361					
24	By Annotator 1 & 2:	6684					
25	Rejected by at least one:	9022					
26							
27	- Averages -						
28	- Decision -						
29	Avg. Kappa Decis:	0.864					
30	- Statistical -						
31	Avg. # of Options:	12					
32	Avg. # of Parses:	91					
33							
34							

List Data

Figure 6.4: Data sheet of the compAnnotLkb report contains all the relevant data extracted from the application including sums used in the computation of the agreement coefficients and lost sentences, as well as partial results for the coefficients.

6.2.4 Application utilization

This section explains how to use the CompAnnotLkb application to compute the Y-Option Kappa (K_Y) inter-annotator agreement metric.

Requirements

The configuration file and each of its fields has already been explained in the Section 6.2.

The CompAnnotLkb was integrally built in Java therefore it is required to have a Java Virtual Machine(JVM) installed of version 1.6 or higher. This virtual machine can be downloaded at `http://www.java.com/en/download/`.

Some external Java libraries are used and are required for the proper usage of the application:

JArgs command line option parsing suite for Java: a library that simplifies the parsing of input parameters and other information.

This allows the application to recieve configuration options by the form of command line flags instead of the configuration file. This can be useful when changing options between executions.

This library can be obtained from: `http://jargs.sourceforge.net/`

Java Excel API: This library allows the manipulation and creation of Excel spreadsheets with Java code and is relevant for the creation of reports and analysis of previous reports.

This library can be obtained at: `http://jexcelapi.sourceforge.net/`

SVNKit: This is a pure Java SVN[4] interaction library. It is only required if using a svn repository for input checkout and output commit or compiling the application.

This library can be obtained at: `http://svnkit.com/`

Execution

The execution of the CompAnnotLkb requires the execution of the following command line in the directory of the CompAnnotLkb.jar:

```
java -cp CompAnnotLkb.jar pt.ul.fc.di.nlx.compAnnotLkb.RunCompAnnotLkb
[-options <args>]
```

The available options are:

h or help: displays this message.

t or temp: working directory where the temporary files and output if no SVN repository is used are stored.

p or pattern: sub-directory pattern to look for within the temporary path location since this tend to change from version to version (regular expressions can be used).

s or source: the path of origin in the SVN repository.

d or destination: the path of the destination in the SVN repository.

r or repName: the svn repository name, if one is used, or project.

o or oldRepName: old svn version repository name.

P or oldPattern: old version parser.

O or oldPath: pattern for previous version sub-directories, analog to pattern for previous version

[4]SVN refers to the Apache Subversion which is a open-source revision control system available at: `http://subversion.apache.org/`

t or tsPattern: primary suite type pattern for identifying main corpus.

A or directoryA: annotator A directory pattern within the temp path

B or directoryB: annotator B directory pattern within the temp path

c or cutPattern: pattern of sub-directories to be ignored during the analysis

1 or annotator1: the name of the directory containing the first annotator [incr tsdb()] log files.

2 or annotator2: the name of the directory containing the second annotator [incr tsdb()] log files.

v or version: version to which the analysis refers to.

The input file should follow the structure presented in Figure 6.1.

If a SVN repository is used, the necessary information to connect to the repository has to be in the svn configuration file, located in:
`pt/ul/fc/di/nlx/compAnnotLkb/util/svn/conf/svn.conf`.
That information is:

REPOSITORY: the URL to the repository to open.

VERSION_PATH: path within the repository (project and subdirectories).

login: login for the repository if required

password: password for the login if required

The information is checked out from the repository location indicated and the resulting report spreadsheet is committed to the repository destination location.

6.3 Summary

The compAnnotLkb is an application fully developed in Java with the aid of the Eclipse SDK.

The main objectives of this application is to aid the process of the LXDeepGramBank annotation task as well as measuring the agreement of the annotators in the previously mentioned task. To this effect the [incr tsdb()] log files are analyzed and compared between annotators and the result is presented in a spreadsheet format report that allows the adjudicator not only to know the value of the agreement between annotators but also the sentences and respective corpus on which they disagreed in their analysis.

Chapter 7

Conclusions and future work

This chapter summarizes the results and conclusions attained during the project documented by this dissertion. Possible solutions for the problems found during the theorization of the metric as well as the development of the application used to compute the metric and the data manipulations needed to achieve this computation.

7.1 Conclusions

During the progress of the documented work there was a necessity to understand the problem and the possible solutions presented by related work and the respective state-of-the-art. This research led to finding a multiplicity of possible approaches, as well as unexpected problems and solutions.

7.1.1 Developed work

The initial part of the developed work went without much surprise by understanding the problem, which included the understanding of the nuclear task of the problem. The initial step on this process was to understand the annotation process and the usage of a double-blind annotation scheme and the work-flow behind the annotation cycle. This scheme of annotation helps to mitigate possible annotator bias in resulting dataset.

The next step was to go through the state-of-the-art and the metrics used in the literature with similar tasks. Unfortunately there is not much work in the area when dealing with this level of detail. Most cases of inter-annotator agreement evaluation are designed and applied to shallow linguistic processing.

There are two main possible paths when devising solutions for the cast with the desired level of granularity: The Cohen Kappa and related metrics that focus on the assignment of categories or decisions upon items; and the Parseval and related metrics developed to classify constituency parsers, by comparing parse results with gold standards.

The annotation task is similar to the one of category assignment but with a much fine granularity, so the best option was to adapt the Cohen Kappa coefficient or one of the inter-annotator agreement metrics related to it, in this case the S coefficient.

The adaptation of the assignment of categories to items was adapted so that annotators decide if the grammatical discriminants available for a given sentence belong or not to the best deep grammatical representation for that sentence. This assignment can be compared, and the degree of proximity measured thus resulting in a granular metric for inter-annotator agreement in the task of corpora annotation with deep linguistic information.

The next step was to explore and understand the LOGON environment.

The dynamic nature of the LOGON environment as well as its main focus on the grammar development, raised some difficulties with respect to the gathering of the necessary information related to the options made by annotators. These problems led to the need of approximation to the theoretical metric model, extrapolating in some cases from the data which was possible to recover from the annotation process. Some data was not stored at all and some information has to be estimated according to the existing data.

An application that computes the coefficient for the data available from the LOGON environmentas as well as the necessary estimates was developed. Adding to this, it also generates a report with the information regarding the data set analyzed and the annotation information recovered.

7.2 Future Work

There is a clear need to either develop an application that aids the dynamic annotation of corpora and takes into account the need to store all the information related to the annotation task. Other option is to adapt the LOGON environment so it stores all possible data. This would greatly help to make the estimates unnecessary, which would permit to obtain an exact calculation of the coefficient.

It would be of extreme interest to apply the Y-Option Kappa metric to other corpora being annotated under the same scheme and with the help of LOGON to cement the results obtained. This was not done during the execution of this thesis mainly duo to lack of time and accessibility of the annotation information of other annotation efforts which usually is not part of the data made available from such efforts.

The dissemination of the metric among people with similar tasks, specially the DELPH-IN members, for comparison of results and metric reviewing will be without a doubt an important part of the future work when it comes to the Y-Option Kappa metric.

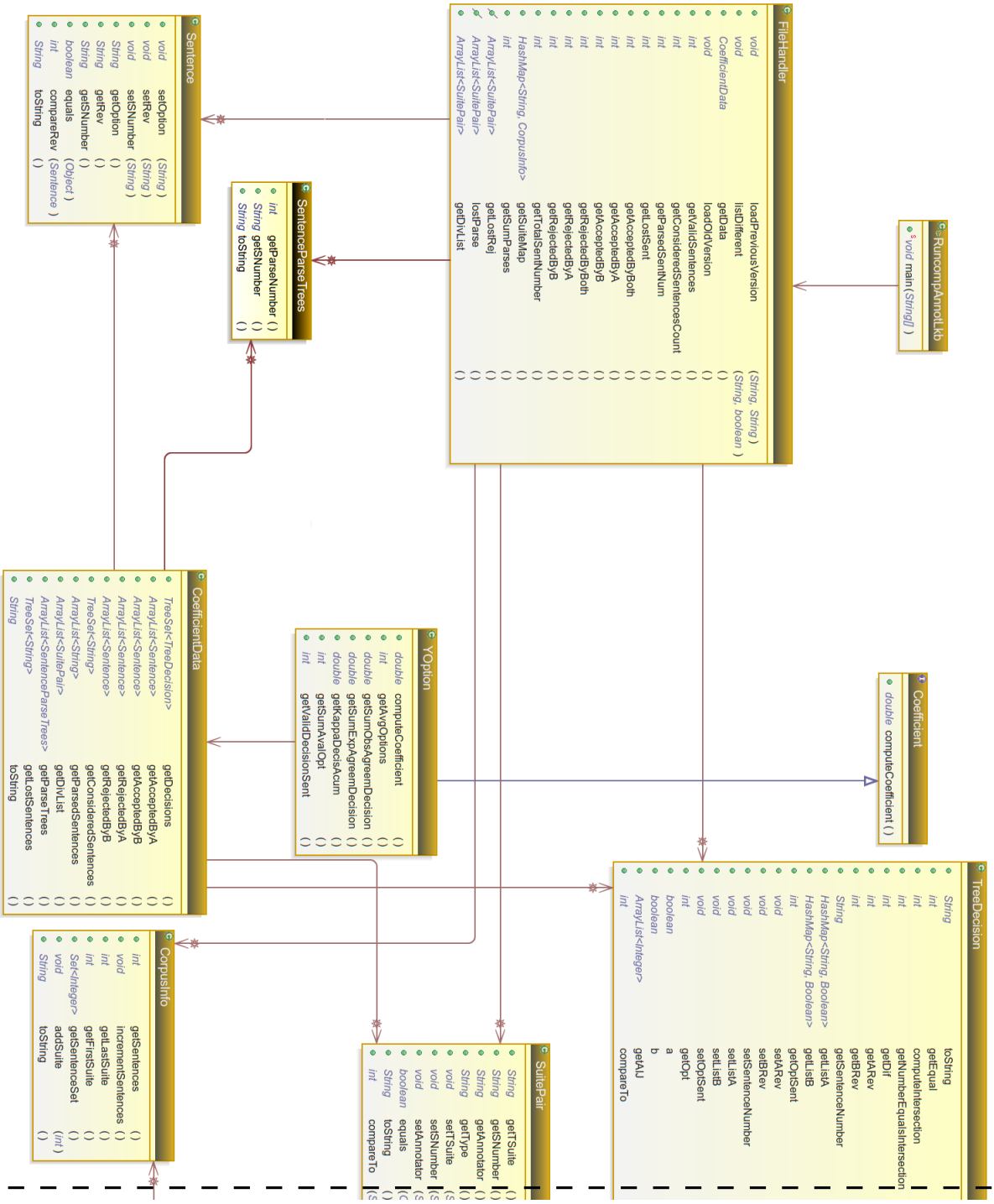
Annex

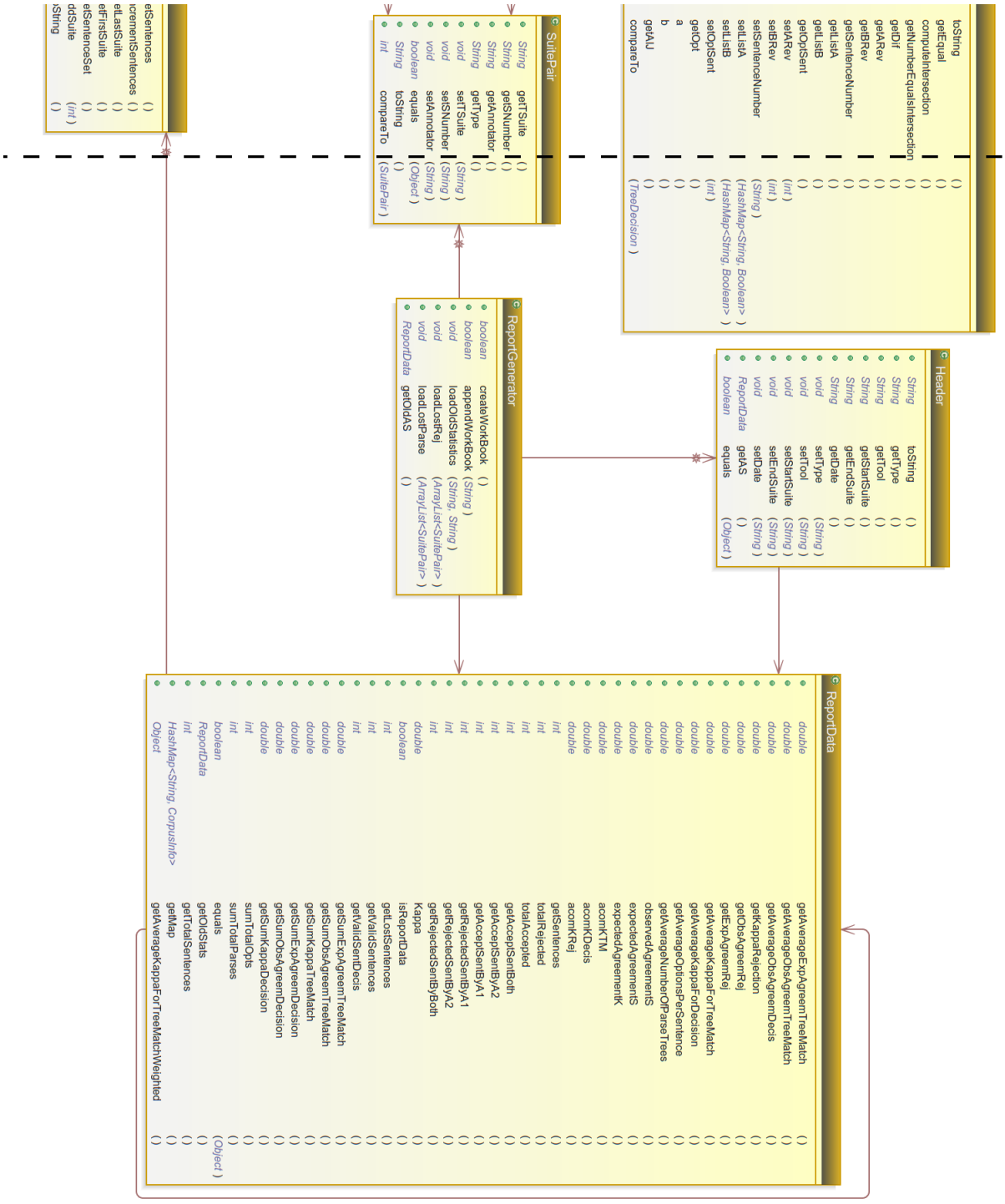
A [incr tsdb()] result log file entry

Bellow is a single grammatical representation entry for the sentence "Foi pura coincidência." ("It was pure coincidence.") in the LOGON result log. Each possible grammatical representation for each sentence has one entry like this one.

8605@0@-1@-1@-1@-1@-1@-1@-1@(129 root 2.71001e+17 0 3 (128 null-subject 2.6652e+17 0 3 (127 head-comp_notclitic 2.52443e+17 0 3 (120 3sg-verb 5.29335e+16 0 1 (119 pret-perf-ind-verb 2.4385e+14 0 1 (4 ser_identity 1.21925e+14 0 1 ("foi" 0 1)))) (126 bare-np -1.25582e+16 1 3 (125 functor-head-hcomps-isect -1.64598e+16 1 3 (122 sg-nominal 1.22622e+16 1 2 (121 fem-nominal 0 1 2 (8 puro_1_adjective 0 1 2 ("pura" 1 2)))) (124 sg-nominal -4.69932e+16 2 3 (123 fem-nominal -5.92554e+16 2 3 (9 coincidência_1_noun 0 2 3 ("coincidência." 2 3)))))))))@@@ [LTOP: h1 INDEX: e2 [e E.MOOD: INDICATIVO E.ASPECT.PERF: - E.TENSE: PRETéRITO-PERFEITO ELLIPTICAL-PUNCT: - SF: PROPOSITION] RELS: < [pronoun_q_rel LBL: h3 [h SCOPE: WIDEST] ARG0: x5 [x PERSON: 2ND-DISTANT_OR_3RD NUMBER: SINGULAR] RSTR: h4 BODY: h6] [pronoun_n_rel LBL: h7 ARG0: x5 ARG1: i8] ["identity_v_rel" LBL: h9 ARG0: e2 ARG1: x5 ARG2: x10 [x NUMBER: SINGULAR GENDER: FEMININE PERSON: 3RD]] [udef_q_rel LBL: h11 ARG0: x10 RSTR: h12 [h SCOPE: NON-WIDEST] BODY: h13 [h SCOPE: NON-WIDEST]] ["_puro_a_1_rel" LBL: h14 ARG0: e15 ARG1: x10] ["_coincidência_n_1-entre-_rel" LBL: h14 ARG0: x10 ARG1: i16] > HCONS: < h1 qeq h9 h4 qeq h7 >] @

B CompAnnotLkb class diagram of the compAnnotLkb application




SuitePair
ReportGenerator
Header
ReportData

Bibliography

Allen, J. and Core, M. (1997). Damsl: Dialogue act markup in several layers. draft contribution for the discourse resource initiative, university of rochester. Available at http://www.cs.rochester.edu/research/cisd/resources/damsl/.

Bennett, M., E., Alpert, R., and C., A. (1954). Communications through limited questioning. *Public Opinion Quarterly*, 18(3):303–308.

Black, E., Black, E. A. S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). Procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 306–311, Morristown, NJ, USA. Association for Computational Linguistics.

Branco, A. and Costa, F. ((to appear)). Hpsg: Arquitectura. In Figueiredo, L., Kelling, C., and de Ávia Othero, G., editors, *Abordagens Computacionais da Teoria da Gramática*.

Branco, A., Costa, F., Silva, J., Silveira, S., Castro, S., Avelãs, M., Pinto, C., and Graça, J. (2010). Developing a Deep Linguistic Databank Supporting a Collection of Treebanks: the CINTIL DeepGramBank. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Branco, A. and Silva, J. (2004). Evaluating Solutions for the Rapid Development of State-of-the-Art POS Taggers for Portuguese. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC2004)*, ELRA, pages 507–510. Springer.

Brants, T. (2000). Inter-annotator agreement for a german newspaper corpus. In *In Proceedings of Second International Conference on Language Resources and Evaluation LREC-2000*.

Callmeier, U. (2000). PET — a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6:99–107.

Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22:249–254.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213–220.

Copestake, A. (2002). *Implementing Type Feature Structure Grammars*. CSLI Publications, Stanford.

Copestake, A. and Flickinger, D. (2000). An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC 2000*, pages 591–600.

Copestake, A., Flickinger, D., Sag, I. A., and Pollard, C. J. (2005). Minimal Recursion Semantics: An introduction. *Jornal of Research on Language and Computation*, 3(2-3):281–332.

Costa, F. and Branco, A. (2010). LXGram: A Deep Linguistic Processing Grammar for Portuguese. In *Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR)*, pages 86–89.

Craggs, R. and Wood, M. M. (2005). Evaluating discourse and dialogue coding schemes. *Computational Linguistics*, 31(3):289–296.

Crysmann, B. (2007). Local ambiguity packing and discontinuity in German. In *Proceedings of the Workshop on Deep Linguistic Processing*, DeepLP '07, pages 144–151, Morristown, NJ, USA. Association for Computational Linguistics.

Di Eugenio, B. and Glass, M. (2004). The kappa statistic: a second look. *Computational Linguistics*, 30(1):95–101.

Dr, P., Uszkoreit, H., Durch, Z., Dr, P., Smolka, G., Callmeier, U., and Callmeier, U. (2001). Fachrichtung informatik universität des saarlandes efficient parsing with large-scale unification grammars.

Dridan, R. (2009). Using lexical statistics to improve HPSG parsing. Master's thesis, Saarland University.

Falk, Y. (2001). *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. CSLI Publications.

Hsu, M., L., and Field, R. (2003). Interrater agreement measures: Comments on kappan, Cohen's kappa, Scott's π, and Aickin's α. *Understanding Statistics*, 2(3):205–219.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree Adjunct Grammars. *Journal Computer System Science*, 10:136–163.

Krippendorff, K. (2004). *Content Analysis: An Introduction to Its Methodology*, chapter 11. Sage, Thousand Oaks, CA.

Lin, D. (1998). A dependency-based method for evaluating broad-coverage parsers. *Nat. Lang. Eng.*, 4:97–114.

Lønning, J. T., Oepen, S., Beermann, D., Hellan, L., Carroll, J., Dyvik, H., Flickinger, D., Johannessen, J. B., Meurer, P., Nordgård, T., Rosén, V., and Velldal, E. (2004). LOGON. A Norwegian MT effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Mitkov, R. (2003). *The Oxford Handbook of Computational Linguistics (Oxford Handbooks in Linguistics S.)*. Oxford University Press.

Oepen, S. (2001). [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany. Avaliable at: http://www.delph-in.net/itsdb/publications/index.html#itsdb.

Oepen, S., Toutanova, K., Manning, C. D., Shieber, S. M., and Flickinger, D. (2002a). Parse Disambiguation for a Rich HPSG Grammar. In *First Workshop on TreeBanks and Linguistic Theories. SOZOPOL*, pages 253–263.

Oepen, S., Toutanova, K., Shieber, S., Manning, C., Flickinger, D., and Brants, T. (2002b). The LinGO Redwoods treebank motivation and preliminary applications. In *Proceedings of the 19th international conference on Computational linguistics*, volume 2, pages 1–5, Morristown, NJ, USA. Association for Computational Linguistics.

Oepen, S., Velldal, E., Lønning, J. T., Meurer, P., Rosén, V., and Flickinger, D. (2007). Towards hybrid quality-oriented machine translation. On linguistics and probabilities in MT. Skövde, Sweden.

Palmer, M., Dang, H. T., and Fellbaum, C. (2007). Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(2):137–163.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Pollard, C. J. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Rehbein, I. and van Genabith, J. (2007). Treebank annotation schemes and parser evaluation for german. In *EMNLP-CoNLL'07*, pages 630–639.

Sampson, G. and Babarczy, A. (2003). A test of the leaf-ancestor metric for parse accuracy. *Nat. Lang. Eng.*, 9:365–380.

Scott, W. A. (1955). Reliability of Content Analysis:. *Public Opinion Quarterly*, 19(3):321–325.

Siegel, M. and Bender, E. M. (2002). Efficient deep processing of Japanese. In *Proceedings of the 3rd workshop on Asian language resources and international standardization*, volume 12 of *COLING '02*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Silva, J. (2007). Shallow processing of portuguese: From sentence chunking to nominal lemmatization. In *Master's thesis, Universidade de Lisboa, Faculdade de Ciências*.

Steedman, M. and Baldridge, J. (2005). *Combinatory Categorial Grammar*. Blackwell. in Non-Transformational Syntax.

Zhang, Y., Wang, R., and Oepen, S. (2009). Hybrid multilingual parsing with HPSG for SRL. In *in Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 31–36, Morristown, NJ, USA. Association for Computational Linguistics.