PE2LGP 3.0: from European Portuguese to Portuguese Sign Language

Rui Jorge Aço Ferreira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Prof. Ana Mineiro

Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia
Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Prof. Hugo Miguel Aleixo Albuquerque Nicolau

November 2018
Acknowledgments

To my father and mother, for supporting my tuitions.

To my advisor, Prof. Luísa Coheur, for having been a remarkable advisor.
To my co-supervisor Prof. Ana Mineiro for her help in the linguistics part of this thesis.
To Prof. Helena Carmo, for annotating the syntax sentences needed for the translation.

To all the amazing friends I met at ISCTE-IUL, for teaching me to help others.
To all of my amazing friends at IST.

Thank you.
Abstract

Since Stokoe stated that Sign Languages are natural languages with rules, plenty of studies have been done for all sign languages. In the computational era that we live in, this means that a merge between computers and sign languages has been attempted, resulting in early stages of spoken languages text to 3D avatar animations’ sign language translation. This thesis is focused on the translation from Portuguese to Portuguese Sign Language (LGP), creating an intermediate sign language representation and testing the avatar animation with JASigning’s avatar. This project tokenizes the input and applies natural language processing techniques in order to prepare the input to follow the LGP grammar and syntax (the syntax is transformed, from Portuguese to LGP, using rules created with the help of an LGP specialist). This part is equivalent to the creation of Sign Tokens. Then, it changes the glosses for their equivalent in SiGML (an XML based transcription language). The result is tested in an existing avatar (JASigning) that directly reads SiGML. The system is evaluated in three different ways: the syntax correctness, the ease of its annotation and the overall correctness of the signs. This thesis presents an early work of an area that still has plenty to develop in the future.

Keywords

Sign Language; Automatic Translation; HamNoSys; SiGML; LGP.
Resumo

Desde que Stokoe disse que as Línguas Gestuais são línguas naturais com regras, imensos estudos foram feitos para todas as línguas gestuais. Na era computacional em que vivemos, isto significa que tem vindo a ser explorada uma cooperação entre as áreas tecnológica e de línguas gestuais, resultando em fases iniciais de traduções de línguas faladas para línguas gestuais, em animações de avatares 3D. Esta tese foca-se na tradução de Português para Língua Gestual Portuguesa, criando uma representação intermediária da língua gestual que contém toda a informação necessária para animar um avatar, mas deixando a própria fase da animação 3D para um trabalho mais futuro. Neste projecto, o input é tokenizado e são aplicadas técnicas de processamento natural da língua por forma a transformar o input e as suas características gramaticais no seu equivalente de Língua Gestual Portuguesa (a sintaxe é transformada, por exemplo, de Português para LGP, através de regras geradas com a ajuda de uma especialista em LGP). Esta fase equivale à criação de Sign Tokens. De seguida, esta informação é trocada pelo seu equivalente em SiGML (uma “transcription language” baseada em XML). O documento SiGML final poderá ser utilizado num avatar 3D já existente (JASigning) que lê SiGML de forma directa. O sistema é avaliado de três maneiras: a sintaxe, a facilidade de anotação de gestos e a exatidão dos gestos. Esta tese apresenta um trabalho base de um campo que ainda possui bastante desenvolvimento futuro.

Palavras Chave

Língua Gestual; Tradução Automática; HamNoSys; SiGML; LGP.
Contents

1 Introduction ............................................. 2
   1.1 Motivation ........................................... 3
   1.2 Goals .................................................. 4
   1.3 Document Structure ................................... 5

2 Background ............................................. 7
   2.1 Sign Languages ....................................... 9
      2.1.1 Sign Language Studies ............................. 9
      2.1.2 About LGP ....................................... 10
   2.2 Computational Approaches ............................ 11
      2.2.1 Machine Translation ............................. 11
      2.2.2 Natural Language Processing ..................... 12

3 Related Work .......................................... 14
   3.1 Transcription Languages ............................. 15
   3.2 Implemented Systems ................................ 17
      3.2.1 International Sign Languages’ Systems .......... 17
      3.2.2 LGP System ..................................... 24
   3.3 Discussion ........................................... 26

4 PE2LGP 3.0 ............................................... 27
   4.1 Architecture .......................................... 28
      4.1.1 Used Transcription Language .................... 28
      4.1.2 Sign Tokenization Module ........................ 29
         4.1.2.A Overview .................................... 29
         4.1.2.B Spacy’s Basics ............................... 31
         4.1.2.C Extracting Morphological Information .......... 32
         4.1.2.D Syntax Dependency Tags ...................... 36
         4.1.2.E Changing the syntax to LGP syntax ............ 37
      4.1.3 SiGML Generation Module ......................... 39
List of Figures

1.1 The first part represents the system that this dissertation implements. The second part is not dealt by this dissertation: instead, an existing avatar is used for testing. 4

3.1 From left to right, an example of the Stokoe Notation and one of SignWriting, respectively. 15
3.2 On the left, Hamburg Notation System (HamNoSys), and on the right, SiGML. 16
3.3 Visual representation of different tracks for each layer in a sign language. 17
3.4 TAG representation of negation in a word manually and non-manually, respectively. 18
3.5 Translation from English sentence to its gloss. 18
3.6 Used representation before changing the value with the appropriate gloss. 19
3.7 Used representation after using the look-up table and switching the english tag with its Newkirk representation of the sign. 19
3.8 Example of Gesture Markup Language (GML) which, like SiGML, is used to represent HamNoSys. 20
3.9 Vietnamese syntax translation proposed. 21
3.10 Example of syntax mapping in IBM Model 1. 22
3.11 In the y axis we can see the input, and in the x axis, the output [1]. 24
3.12 Architecture used in [2]. 24
3.13 Example of struct used in [2]. 25

4.1 Visualization of the project’s architecture, from input to output. 28
4.2 eSIGN’s window. 29
4.3 Overview of this part of the thesis. In the end we have the input in Língua Gestual Portuguesa/Portuguese Sign Language (LGP) syntax order, classifiers and Non-Manual Expressions, all of which together correspond to Sign Tokens. 30
4.4 Representation of Sign Tokens. They appear in LGP’s syntax’s order, contain morphological information about them and can be later easily read in the SiGML Generation Module. 32
4.5 Picture that depicts the contents of a doc variable, given a certain sentence as its input. 33
4.6 A snapshot of a line that depicts a possible full tag of a verb. As we can see, one of the annotations is `<mv>`, which stands for main verb, telling us that the token from which this tag is taken from is the predicate. .......................................................... 33

4.7 A snapshot of a few lines that show full tags. The first morphological tag of each line contains information about the word’s main morphological class (in this case, `ADJ`, `PRP`, `ADJ` and `DET`, respectively). .......................................................... 34

4.8 First, the sentence *O João come a sopa.* is shown, and then, its corresponding syntax dependency tags (as provided by Spacy). .......................................................... 36

4.9 In this picture, we can see that the *ROOT* is, in fact, *come*, as it is visible that all of the arrows start from that same token. It is also visible that *João* is the subject to *come*, and that *sopa* is the direct object to *come* as well. .......................................................... 37

4.10 This Figure shows a few of the used sentences and the resulting annotation made. .......................................................... 38

4.11 Overview of the second part of the thesis. The system receives the *Sign Tokens* and outputs a file with the SiGML necessary for the 3D animations. .......................................................... 40

4.12 Overview of the part of this module that deals with the creation of our SiGML dictionary. .......................................................... 41

4.13 An example of a SiGML file. It has the opener and closer tags on the first and last line, and the letter *J*’s equivalent in HamNoSys (in order to give a simple example). .......................................................... 42

4.14 A print-screen of some sentences, input and output. Just with three sentences, we study the difference between past-tense of the verb and interrogation. Note: ENM stands for Non-Manual Expression. .......................................................... 43

4.15 The annotated sentence was the original sentence used to create our rule. Afterwards, we can apply this same rule to other sentences with the same syntax, just like Possible Input 1 and 2 in this Figure. .......................................................... 44

4.16 We create a rule after looking at the annotation: *PREDP-[obj]* represents the verbal classifier between that verb and object. .......................................................... 44

4.17 ........................................................................................................ 45

4.18 An example of an exception. As can be seen, the rule itself is only applied when a specific sentence is found on the input. .......................................................... 45

4.19 Revision of how the system’s dictionary is generated. .......................................................... 46

4.20 Pattern of a HamNoSys line. .......................................................... 46

4.21 Snapshots of *SpreadTheSign’s sopa* annotated video for LGP. .......................................................... 47

4.22 Depiction of how to separate the dominant hand from the non-dominant hand in HamNoSys writing. This applies to configuration and orientation sets, and also to location and movement sets of characters. .......................................................... 47
List of Tables

4.1 Table depicting Spacy’s Part-of-Speech (POS) tags and their equivalents in Portuguese grammar. .......................................................... 33
4.2 Table depicting Spacy’s dependency tags and their equivalents in Portuguese Syntax. ... 36
4.3 Table depicting the verb tags that were created to associate to the syntax exchange rules. 38
C.1 Table depicting Spacy’s tag map morphological classes. There are a few more, but they are duplicates without being in capital lettering (for instance, besides ADJ there is adj). The question marks (marked as ? [?]) represent unknown meanings: these were classified as such only after not finding any guides or e-mailing Spacy’s team on these meanings and not getting an answer back. ......................................................... 81

List of Algorithms
Listings

4.1 Syntax rule that is applied to O João come a sopa.'s syntax structure (more precisely, João come sopa.: this is because some tokens - articles - are removed (see Section 4.1.2.B)). A delimiter (→) is used, dividing the input from the output. .............................. 30

4.2 Syntax rule that is applied to O João come a sopa.'s LGP syntax structure (more precisely, João sopa come). Afterwards, it becomes João sopa come[sopa] .............................. 39

4.3 Stored Sign Tokens' format. .................................................. 40

A.1 Code used in order to load Natural Language Tool Kit (NLTK)'s already existing sentence segmentation model. .................................................. 71

A.2 Line of code that defines the model that Spacy will be using. .............................. 72

A.3 Code that returns the tags between < and >. ........................................... 72

A.4 Code that returns the tags not in between < and >. ........................................... 73
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASL</td>
<td>American Sign Language</td>
</tr>
<tr>
<td>LGP</td>
<td>Língua Gestual Portuguesa/Portuguese Sign Language</td>
</tr>
<tr>
<td>CSE</td>
<td>Czech Sign Language</td>
</tr>
<tr>
<td>SC</td>
<td>Signed Czech</td>
</tr>
<tr>
<td>VSL</td>
<td>Vietnamese Sign Language</td>
</tr>
<tr>
<td>BSL</td>
<td>British Sign Language</td>
</tr>
<tr>
<td>LSF</td>
<td>Langue des Signes Française/French Sign Language</td>
</tr>
<tr>
<td>GSL</td>
<td>Greek Sign Language</td>
</tr>
<tr>
<td>DGS</td>
<td>Deutsche Gebärdensprache/German Sign Language</td>
</tr>
<tr>
<td>ISL</td>
<td>Indian Sign Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>SWML</td>
<td>SignWriting Markup Language</td>
</tr>
<tr>
<td>GML</td>
<td>Gesture Markup Language</td>
</tr>
<tr>
<td>HamNoSys</td>
<td>Hamburg Notation System</td>
</tr>
<tr>
<td>MT</td>
<td>Machine Translation</td>
</tr>
<tr>
<td>POS</td>
<td>Part-of-Speech</td>
</tr>
<tr>
<td>NLTK</td>
<td>Natural Language Tool Kit</td>
</tr>
</tbody>
</table>
1

Introduction

Contents

1.1 Motivation ........................................... 3
1.2 Goals ................................................. 4
1.3 Document Structure ................................. 5
1.1 Motivation

According to the World Federation of the Deaf, about 70 million people use a sign language as their main source of communication. Just this number alone should give the reader a feeling about the importance of studying this field. However, its study is not as straightforward as one might think: not only are sign languages not yet fully understood, there are also many sign languages available and they are all different. This contributes to the fact that, computationally speaking, a system that is able to process all of the sign languages in a generalized manner is not possible to achieve. In general, there are sign languages for each country (or region), ranging from American Sign Language (ASL), eventually to our own Portuguese sign language, Língua Gestual Portuguesa/Portuguese Sign Language (LGP). In addition to this, sign languages evolve with time, just like spoken natural languages (for instance English and Portuguese).

All of these sign languages share a common problem: none have its official writing system, even though there is a strong attempt in Brazil to make SignWriting its official writing system. This means that if we somehow want to write a sentence using sign language and in an internationally (or nationally) accepted manner, we cannot. The advantages of having a writing system for sign languages are many, being one of them providing a way for the deaf to write and read sign language. One way to get the most out of a writing system of sign languages has been studied significantly in the last decades: automatic translation of an oral language to a sign language and, with the result, converting our text representation of the sign language into 3D animations, with the help of an avatar (i.e: write something in a spoken language and have an avatar translate it to signs of a certain sign language).

If the reader is not familiarized with how sign languages work, then it might be difficult to visualize how hard this translation actually is. Sign languages not only require the use of both hands, but they also require the use of non manual elements, like facial expressions, including the use of the eyebrows, mouth and nose or body tilting. The good news about all this is that there is plenty of recent related work, as the combination between sign languages and machine translation has been a hot topic over the recent past.

There are many advantages as to why we would want an automatic translation from a text in an oral language to visual rendering of that content in a sign language (by an avatar), namely:

• the power to turn any given text, written in an oral language, available to the deaf community;

• avoiding awkward situations where the human translator needs to translate something on a more
uncomfortable subject (sexual education, for instance).

Besides these two, there is also a language specific advantage: a considerable amount is done for English, but not as much for Portuguese. In fact, despite some attempts, LGP’s automatic processing did not have as much coverage as other sign languages within this area, yet making it a fresh field to start studying.

1.2 Goals

A great sum of the work that has been done in this area tried to do it all: from translating a text to a certain format that tries to hold the information needed from a certain sign language (henceforth called Sign Language Intermediate Representation), and using all of this data to somehow move an avatar’s body, thus creating our translation via an avatar [3].

This dissertation does not focus on the creation of an avatar movement generation part, and develops an automatic translation system from European Portuguese to LGP, provided that the translated outcome (in the format of a Sign Language Intermediate Representation) is ready for future use by an avatar (Figure 1.1). Instead of creating an avatar from the beginning, this dissertation’s project uses an already existing avatar in order to test its translations (this avatar is JASigning ²). This means that instead of working on the whole process, this dissertation is focused on the part before having an avatar playing the translated sentences/signs in LGP. More specifically, it deals with the natural language processing of any input sentences in oral European Portuguese, translating it into a text format intermediate representation of what is being translated, in LGP. The avatar JASigning is used to test this Intermediate Representation.

Figure 1.1: The first part represents the system that this dissertation implements. The second part is not dealt by this dissertation: instead, an existing avatar is used for testing.

Figure 1.1 shows us that the dissertation only implements the translation from Portuguese to LGP up to a text format (an intermediate representation). The process of this implementation includes changing the syntactic structure from oral Portuguese to LGP, since these are different languages and possess different syntax. Then, given the information we gather in all of the past work in this area, we use a tran-

²http://vh.cmp.uea.ac.uk/index.php/JASigning
**scription language** (a language that holds information on signs, like its movement, location, etc.) that fits best for our specific situation (this dissertation uses **Hamburg Notation System** (HamNoSys) [4]), taking into account the language we are dealing with and our needs, depending on the grammar rules, morphology, etc. **SiGML** is an Extensible Markup Language (XML) representation of HamNoSys, and it is used in order to leave the output readable by an avatar. Finally, when the output is in SiGML, it can be tested using JASigning.

In this thesis, we not only explore the possibility of creating a system that translates text from an oral language to a sign language, but also of creating a system that is self-sustained by people annotating vocabulary and syntax rules, allowing people to populate it with data. Can people annotate a sign language translation system with vocabulary with ease, and is it a feasible process? Does the choice of the system’s transcription language take this ease of annotation into account? The answer to these questions may be a solution against the lack of LGP data.

Besides what was previously stated, in this thesis we study certain aspects of LGP syntax, and with it apply rules that are used for a better coverage of the problem of word alignment. Can the usual syntax structure used in Portuguese European to LGP projects of **Object – Subject – Verb** (Section 2.1.2) be extended to more translatable syntax structures in our system?

### 1.3 Document Structure

The following list names the remaining structure of the document, as well as a brief description of what is written in each section:

- **Background (Section 2)** provides a background for the reader to get familiarized with elementary, yet needed, information on sign languages both in general and specifically for LGP, and on machine translation;
- **Related Work (Section 3)** references work that has been done in the past within this area, both with LGP and other sign languages in the world, providing methods that have been used and that can be reused to achieve our purpose;
- **PE2LGP 3.0 (Section 4)** explains the thought process behind the implementation of the PE2LGP 3.0 system;
- **Creating a Dictionary - eSign (Section 4.3)** explains how our dictionary is created writing in HamNoSys and getting SiGML, through the use of a software called eSign;
- **Evaluation (Section 5)** evaluates our system appropriately and gathers the results;
- **Conclusions and Future Work (Section 6)** presents a brief conclusion on what is to be done in the
future and on what can be achieved with this solution.
2

Background

Contents

2.1 Sign Languages .................................................. 9
2.2 Computational Approaches ................................. 11
This section is divided into two main subsections:

**Section 2.1 - Sign Languages** presents a historical point of view to the reader, showing how sign language was first identified as a natural language, and the evolution of its study. Then, some basic notions on how sign languages work are also pointed out, including rules that apply in general (as well as specific ones for LGP);

**Section 2.2 - Computational Approaches** explains machine translation concepts that are applied in the task of translating from an oral language to a sign language.

# 2.1 Sign Languages

## 2.1.1 Sign Language Studies

Before the 60’s, not much had been studied about sign languages. In fact, sign languages were not even considered natural languages at all. The first man to recognize sign languages as natural languages was William Stokoe [5]. In his work, contrary to what people believed before, Stokoe said that just like oral languages, sign languages possessed minimal pairs. He created the Stokoe Notation, being this the first attempt at textually representing the ASL.

Meanwhile, more has been studied and some conclusions are agreed upon everywhere. Phonology is related to the organization and construction of larger elements through smaller elements, and it studies the phonological relationships between these in a language [6]. This language (when we talk about phonology) used to be considered only a spoken language, until Stokoe presented his studies, and introduced the world to a new kind of phonology, specific to sign languages. In fact, it was introduced as cherology, but it was changed to phonology, as a means to intuitively demonstrate the resemblance between the structure of both modalities of languages (oral and signed). In a spoken language we can define a phoneme as a single unit that is part of the construction of words. A good analogy lies in the use of Lego, where we use singular pieces to create unique combinations. Let these pieces be our phonemes. These pieces differ in shape and size, and if we change one of these specifications, it can entirely change the looks of our work of art. With signs, we also have phonemes (formerly known as cheremes), which are separated into its configuration, localization, movement and orientation. These are sign primes, and by changing one, we are changing the sign itself. Besides these, every sign language also possesses non-manual elements. The manual elements are the ones performed with our two hands. The non-manual elements, however, deal with other means of communicating other than with the hands, like using facial expressions, shoulders, neck and even gaze. These

---

1The first attempt to represent any sign language was made by L'Epée in the eighteenth century, to represent his signes méthodiques. [5]

are just as important, since in LGP they can also entirely change the meaning of a sign.

2.1.2 About LGP

The first official study for Portuguese Sign Language was achieved in 1980 by Maria Isabel Prata and Raquel Delgado Martins [7], and it was considered our first dictionary for LGP. Eventually some studies were made on LGP [8–11], but nothing compared to the likes of ASL, making LGP a fresh field for studying. It is of the utmost importance to try and understand the many characteristics of the Portuguese Sign Language. The difference between LGP and other signed languages lies in the rules that are applied in it, whether in its phonology, morphology or syntax.

The Portuguese syntax follows a Subject - Verb - Object (SVO) structure, while LGP’s syntax generally follows an Object - Subject - Verb (OSV) structure. However, there is plenty of LGP grammar that we do not know of. Besides this, Portuguese is also made up of other syntactic functions, such as, for example, complements. LGP’s grammar only states that it generally follows an OSV structure, and does not mention the other elements that can be found in a sentence.

After syntax, we go to morphology. Additive morphemes occur when we add an affix to our sign [12]. In sign language, these are very common, and are used to change the context, add spacial information, define the intensity of the sign, etc. Some important ways we can use additive morphemes in LGP are:

- **Negation**: we can simply sign NO after our sign, but we can also use non-manual elements in order to do so (our head) or moving our hand down and to the outside repeatedly;

- **Grammatical Facial Expressions**: these define the grading of the names, as in a little concerned or very concerned, for example. these also state the type of sentence (declarative, imperative, interrogative, etc.);

- **Intensity (of the sign)**: if we do not use facial expressions for this, we can also use, for instance, the amplitude of our hand movement. A good example is moving our hand a short distance for the sign FAR AWAY and moving it a long distance for VERY FAR AWAY [13].

Morphemes can be repeated or duplicated. Repeated morphemes are, like the name suggests, repeated in time, while duplicated ones are also done with the other hand, but not repeated. These are important concepts because plurality of names can be achieved through repeated morphemes. This is the case for the words tree and trees. For the singular, we simply perform its sign. For the plural, we
do the same but while shaking our hand back and front, in order to indicate repetition. This can also be
done with duplication, but only when a sign made for the singular already uses repetition in itself.
Some exceptions include riding a bike or riding a car. The sign for riding will be different in each of these
examples, depending on the vehicle. These are alternative morphemes.

In sign languages we also have classifiers. These are morphemic elements that are similar to signs,
but in reality they are used to describe or qualify people, animals, things (like objects) and actions [11].
For a person who is not a linguist with knowledge on sign language, this definition will be confusing.
Informally, classifiers are identical to signs, but they classify a group of some reference. Classifiers exist
in almost all sign languages, but they are/may be different in each sign language. Classifiers can be
nominal or verbal [13]:

Nominal Classifiers can be classified into descriptive, attributive and specifiers. Descriptive classifiers
are used for the representation of feelings, places, surfaces and groups of elements of Nature
like humans and plants. Attributive classifiers, like the name suggests, are used for defining attributes like size, shape, texture, smell, and so on. Finally, specifier classifiers are used to describe
a trajectory, a direction, a location or the disposition in which the references are set in.

Verbal Classifiers represent specific situations where in order to connect a specific combination of
verb with subject, or with an object, or to show the way it happens visually, or even the location, the
sign itself is exceptionally changed. In order to understand better, please take into account these
two sentences: “John eats the soup.” and “John eats the bread.”. There is a generalized way to
sign the verb to eat, but in reality, the way one holds the food while eating depends on what one is
eating. So, when gesturing “eats the soup.”, the sign changes, making the secondary hand a bowl,
and eating from it. Another example that is easy to understand is hand-writing as opposed to writing
in a keyboard. Finally, two verbal classifiers cannot occur simultaneously.
In LGP, to sign names of people, like John, gestural names are used. This only applies, however, if
a person has one associated with him/her. When there is no gestural name associated with a person,
his/her name is spelled letter by letter. For example, if a person named John does not have a gestural
name, the letters J, O, H and N are spelled individually.

2.2 Computational Approaches

2.2.1 Machine Translation

The problem that we are faced with in the entirety of this project is to automatically translate a language
to another, without us manually interfering in this process (i.e: only the computer is involved in the
process). This type of automatic translation is also known as Machine Translation (MT), and started being discussed in the 50’s [14]. Mainly two types of MT are used in this specific situation (oral to sign language translation): Rule-based and Statistical MT. Neural MT also exists, but it has not been particularly popular among this specific type of project.

**Rule-based MT:** in this approach rules on the source and target languages’ syntax, morphology and semantics are created. Then, whether through a direct or through an intermediate transfer based translation, and with the help of a lexicon, the translation is performed. For the task of translating a text into a signed language, an intermediate text representation is a common approach. The input is text and the output is defined by movements shown with a 3D avatar, so a direct translation (without an intermediate text representation) is not something feasible;

**Statistical MT:** within a statistical approach, we need a large amount of bilingual corpus. This type of translation follows statistical models, but the problem is that a large corpus is not always available. This is the case for signed languages outside of English speaking countries, like Portugal.

### 2.2.2 Natural Language Processing

Natural Language Processing includes a wide variety of techniques used to process ‘naturally occurring texts’ [15]. This includes looking at things like morphology, syntax and semantics.

In Section 2.2.1, we mentioned that in order to use a statistical approach to MT we need a large corpus. *Corpus* is a large collection of data that allows us to draw statistical models from it. Because we are dealing with a translation problem which starts with text in portuguese and ends in an sign language intermediate representation, if we were to take a statistical approach to our translation, our corpus would contain sets of text in Portuguese. In order to analyze the syntax of such corpus, we need to annotate it. This can be achieved with Part-of-Speech (POS) tagging, which tells us each word’s morphosyntactic category.

Some sets of corpus that are completely annotated are available. These are the so called tree-banks. A very complete library that allows us to work on Natural Language is the Natural Language Tool Kit (NLTK) [3]. From it, we can access plenty of corpora, including the Portuguese treebank: Floresta Sintática [4]. Another great library for Natural Language Processing is Spacy. Spacy is great since it also allows us to extract syntactic dependencies directly. Spacy is used in this thesis.

Corpora are very important in MT. Unfortunately, it is an extremely time-consuming job to build one,

---

and because many times one is not available, it leads to projects taking a rule-based approach \([2,3,16]\). This is the case of this thesis: our translation system adopts a rule-based approach.
3 Related Work

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Transcription Languages</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Implemented Systems</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Discussion</td>
<td>26</td>
</tr>
</tbody>
</table>
This section will explore some of the most important related work done on Text-to-Sign Automatic Translation systems. First, available and popular transcription languages are presented, and then, projects of spoken language to sign language translation are explained, with advantages and disadvantages being pointed out along the way.

### 3.1 Transcription Languages

Since the 60’s, many were the attempts at representing sign languages in text. Many of these are represented through icons that try to simulate and show what the sign actually looks like, and all of its elements. These are quite more readable to the human eye, but in terms of computer processing, they are not as straightforward to process (a great example is HamNoSys [4]). Other attempts are the so called glosses, which are represented by the letters of our own alphabet. These are obviously easier to process, but harder to understand, since unlike icon representation, these do not give away anything visual about the sign it is representing (an example is the Stokoe Notation [5]).

The first attempt at a notation system for a transcription language was the Stokoe notation, created by William Stokoe himself [5]. As referred in Section 2.1.1, Stokoe defined each sign to be made up of a **handshape** (configuration), a **location** and some **movement**. An **orientation** was added later as part of a sign too.

Later in 1974, SignWriting was created [17]. It is made up of created symbols to represent sign language, and unlike the Stokoe Notation, it allows for the representation of non-manual elements. Each **gloss** written using the Stokoe Notation is made up of three characters: one for location, another for handshape and yet another for movement. However, SignWriting uses only one symbol per sign. Each symbol tries to visually emulate a sign entirely: both manual and non-manual elements (as seen in Figure 3.1).

![Figure 3.1: From left to right, an example of the Stokoe Notation and one of SignWriting, respectively.](image)

As SignWriting tries to emulate the symbols themselves, as long as a deaf person learns a sign language textually with SignWriting from the start, it is a learnable and easy way of textually representing it. It is actually seen as a **movement writing system**, given that its representation is fluid and appealing.
to the deaf community, in general [17]. As far as processing goes, since SignWriting uses symbols, it needs a different representation that enables its computational processing. Special instances of XML allow this to happen. XML is a meta-language that serves as a data holder for any data that we want to store \(^1\). Due to its flexibility, XML is usually thought of as a practical solution for this kind of problem. It was certainly the answer to SignWriting’s problem, as SignWriting Markup Language (SWML) was created. SWML can be used to represent each sign into a set of its features, according to SignWriting.

Another very famous attempt at sign language representation is the HamNoSys. It combines both the Stokoe Notation, in the sense that each sign is represented by handshape, orientation, location and actions, and the SignWriting, in the sense that these are represented by icons. Contrary to the Stokoe Notation and SignWriting, it was not made for the common reading and usage from deaf people: its creation was for academic purposes, and it is a widely popular choice. Just like SignWriting, computationally, HamNoSys can be represented by XML based languages. HamNoSys was used in this project and the used markup-language was SiGML. An example of both can be seen in Figure 3.2.

![Figure 3.2: On the left, HamNoSys, and on the right, SiGML.](image)

Although the previous options for representing sign languages are acceptable, they treat signs equally in time. This means that after a sign, a transition period comes, and then another sign, and another transition period, etc. It works in sequentiality. This is an approach that works for simpler cases. However, this means that any non-manual element that we try to represent in our avatar will always be associated with a sign in specific. The truth is that sign language is multimodal: unlike our oral languages, which letter by letter are defined as sequential, sign languages work more as different layers at the same time, and one on top of the other. Having non-manual elements being reproduced independently of when a manual element starts makes everything more natural and understandable. Filhol knew this, and created AZee [18].

\(^1\)https://www.w3.org/XML/
Filhol compares his take on this problem with computer animation, which uses key frames. When we consider a timeline, at different t’s (times) we can create key frames and associate a specific animation to each. Then, an interpolation between each pair of key frames occurs. Instead of using computer animations, Filhol used this method to create a new transcription language: the Zebedee [19]. Moreover, like in video editors we get to manage different timelines (in the form of video and audio tracks) for different layers, we may also want to separate different elements in signing to different layers, so that they can operate independently in time. So, just like video editors, we can also have different tracks (Figure 3.3), for instance having different tracks for the facial expressions and the hand movement. Filhol created this concept and called it Azalee. Together, Zebedee and Azalee merge into a new transcription language called the AZee.

![Figure 3.3: Visual representation of different tracks for each layer in a sign language.](image)

AZee treats Time Interval units (each Time Interval indicates the period over which a sign is active) with Allen’s precedence time operators. This allows for simplicity when defining the placement of each element in its timeline: instead of giving an exact time value for when it should be signed, we just represent when it should be signed relatively to the main sign.

### 3.2 Implemented Systems

#### 3.2.1 International Sign Languages’ Systems

TEAM (Translation from English to ASL by Machine) [3] was the first attempt at fully translating English into an intermediate language and then using it to create 3D animations to represent it. Its intermediate language used glosses alongside some other information that came embedded to it, like some non-manual elements. This project came as the “eye-opener” of this whole idea (translating a spoken language to a sign language), demonstrating that this kind of project is not only doable but also worth developing further. It was made for ASL, but it also allows for the processing of other languages. The movement of the avatar works by keypoints. In this work, there are two kinds of keypoints: Goal keypoints and Via keypoints. Each type of keypoint serves its purpose:

- **Goal Keypoints**: these define a target location, to where the hand is moving to;

---

2[https://en.wikipedia.org/wiki/Allen%27s_interval_algebra](https://en.wikipedia.org/wiki/Allen%27s_interval_algebra)
• **Via Keypoints**: these represent locations where the movement must go through to reach certain Goal keypoints.

These keypoints are all relative to the shoulders of the avatar model, and the locations defined by the keypoints are used by the wrists, leaving also room for its orientation.

As for the translation itself from English to *glosses*, a Lexicalized Tree-Adjoining Grammar [20] is used. Rules that are known to apply for ASL are used to change the syntax from English to ASL, given that typically ASL uses the SVO order (Subject - Verb - Object), which facilitates things, since it is the same as in English. Most of these rules may be irrelevant to this project, since we need to apply rules from LGP and not ASL. We can, however, keep these in mind and maybe use it for our own purpose. An example is the rule that applies for the **negation** of words, which can be made manually or non-manually (as a sign or as a non-manual element), as shown in Figure 3.4. In LGP, we can also add the sign No to what we want to negate, or use head movements [13].

![Figure 3.4: TAG representation of negation in a word manually and non-manually, respectively.](image)

The last aspect from this work that may be interesting to emulate is the way it treats, for instance, adverbs such as the one demonstrated in Figure 3.5. The adverb *slowly*, instead of being translated into its own *gloss*, is added as embedded information within the *gloss* of the word that modified this adverb. In Figure 3.5, *slowly* refers to the verb “open”, which means that the information from *slowly* is added to the verb. This information includes four parameters that represent *Space, Weight, Time* and *Flow*. These range from -1 to 1, and with the right values we can sign the OPEN word as intended (slowly).

![Figure 3.5: Translation from English sentence to its gloss.](image)

In 1999, there was an attempt [21] to fully translate weather reports from the National Weather Service from the United States, gathered from the World Wide Web. Angus Grieve-Smith, the author, chose to do so in order to work with a more closed domain, limiting the used vocabulary to the typical words that we hear in a forecast. This was achieved using Newkirk’s *literal ortography*, which uses the roman alphabet. This translation method did not take into account non-manual expressions and the movement
of the fingers, so it was added in Grieve-Smith’s project. For each word that was parsed from the English original forecast, it was put together alongside a tag that represents the lexical value of the word. As the author exemplifies, TODAY is changed to TODAY <DAY>, or WITH to WITH <PREPOSITION>. When a set of words means something that can be translated as a single word, the tag represents this word. This is the case with RAIN SHOWERS, which is pretty much <PRECIPITATION>. Then, with his own format, the tags are stored (Figure 3.6). With the help of a lookup-table, the tag values are changed for their respective values written in the literal orthography of Newkirk (Figure 3.7).

\[
[1][1]{windP}\{time\}[1]{time}=afternoon
\]

**Figure 3.6:** Used representation before changing the value with the appropriate gloss.

\[
[1][1]{windP}\{time\}[1]{time}=byy:yd
\]

**Figure 3.7:** Used representation after using the look-up table and switching the English tag with its Newkirk representation of the sign.

Although interesting, this project serves as an example of what we do not want to replicate: it works, but it does not look simple to learn. The reason for this is that the notation is not obvious, and we want to use or recreate something that both works and is easy to use/read, as well as annotate.

Another project that attempts to translate a spoken language to a sign language is the Czech-Sign Speech Corpus for Semantic based Machine Translation [22], translating Czech to Czech Sign Language (CSE). CSE is, just like LGP or ASL, a language for the communication between deaf people alone, but it is interesting to mention that the Czech also have a language for the communication between the deaf and the hearing: the Signed Czech (SC), and this is what they translate their utterances to. They wanted to tackle the problem of statistical oriented machine translation doing this kind of work, which is the need for large corpora, and so they created one with train timetable dialogues from customers. Then, they annotated it within three dimensions:

- **Domain:** the first dimension tries to set the utterance onto a given area of what kind of spoken conversation is happening (ex: an apology, or a task);
- **Speech Act:** specifies the client’s request (ex: request-info, status-report)
- **Semantic:** it gathers the needed information to answer the client’s question, such as time of departure, destination, etc.

For the translation process, they used a dictionary with around three thousand Czech signs, and they added a special sign for *when a word is not correctly translated to its sign* (an interesting approach
to not being able to translate some word, given that by context, mayhaps a deaf person can still understand the full sentence, even if missing a word from it). In other words, when a word is not found in their dictionary, they use this special sign to represent that a word was not found, and this way, a partial translation is still achieved. They used direct translation, given that when a word with more than one translation was found, they would choose the most likely translation. The sentence error percentage was of 50.5%.

One of the most successful projects in this matter was the ViSiCAST project, from 2000 [23]. ViSiCAST tries to translate English to sign language through the usage of a custom markup-language, much like XML: the Gesture Markup Language (GML). GML is built over HamNoSys, and represents a sign through glossing, its phonology, its phonetics and articulation. So English text is given into a GML parser, which is then put into the avatar to play the animations. An example of GML is given in Figure 3.8.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE gml SYSTEM "gml.dtd" >
<gml>
<avatar url="Tessa ava" id="A" alt="Tessa" />
<sign glose="TO-AND-FRO">
<hamnosys>
    <righthandgesture>
        <handshape form="fist" thumb="across" /> 
        <handlocation where="shoulder" offset="rightOf"/>
        <handorientation extfinger="upN" palm="down"/>
        <movement direction="horW" repetitions="repeat1" repeatmode="fromstart" />
    </righthandgesture>
</hamnosys>
</sign>
</gml>
```

**Figure 3.8:** Example of GML which, like SiGML, is used to represent HamNoSys.

On more recent studies, a project whose aim is to translate Vietnamese to Vietnamese Sign Language (VSL) [16] tries to engage in the problems that we also face in this thesis: a language’s syntax, phonology and morphology. Apart from their final translation approach, it is crucial to verify that the first problem they tackle is the syntax: the order of the tokens, negations, numerals or question sentences. For all of these things, they analyse examples in Vietnamese and examples in VSL. Then, with all this information, they use rule-based syntax trees to change the syntax from Vietnamese to VSL. They take an interesting approach (Figure 3.9) on the translation itself, where first they label all the words with their syntactic functions, and if they cannot label a word, they try to find a synonym and label the initial word.
with the synonym's syntax role. When all is labeled, they find an already existing sentence with the same structure as the one they are transforming, and change the syntax accordingly. This is a very promising approach, as it scored 97.5% of success in its evaluation, using BLEU \cite{24}, on 200 test sentences.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure3_9.png}
\caption{Vietnamese syntax translation proposed.}
\end{figure}

As far as statistical translation methods used on the exchange between the source and the target language go, Achraf Othman and Mohamed Jemni \cite{25} present some interesting approaches. As a base for their work, they possess corpora with only English sentences and their matching sentences in ASL. As previously said, the problem with all statistical approaches from spoken to signed languages is that if we want to use it, we need large amounts of corpora from the signed language we are aiming at, and for most, this still is not a reality. So, in general, for all attempts made with a statistical approach, corpora has been mandatorily created. This can be extremely time consuming. At this part, IBM Model 1 comes along. Othman and Jemni's approach on this work starts with the use of a dictionary between source and target words. We can have two cases of matching words:
• **One Source Word and One Target Gloss:** we just make the change between them;

• **One Source Word and Many Possible Target Glosses:** in this case we have to use statistics.

In the previous second option, one way we can do this is by counting the words in our corpora, and the one with the biggest count can be our chosen gloss. If we have a word $f$ in English and a gloss $g$, the probability of getting $g$ provided that $f$ is given by: $P(g|f)$.

We then have an initial sentence, and a transformed sentence made with their respective glosses, but the syntax may not be correct: we still need to change the order of the glosses according to our sign language syntax rules. In model 1, this happens through the following mapping: $j: a:j \rightarrow i$. What this means is that each word is initially in a certain position $j$ and ends up in another position $i$, as exemplified in Figure 3.10.

$$a: \{1 \rightarrow 1; 1 \rightarrow 2; 2 \rightarrow 3; 2 \rightarrow 4\}$$

**Figure 3.10:** Example of syntax mapping in IBM Model 1.

Given that $l$ represents the length of a sentence, $f$ an English sentence and $e$ a sign language sentence, the function used for this word alignment is:

$$p(e, a|f) = \frac{\epsilon}{(l_f + 1)^l_e} \prod_{j=1}^{l_e} t(e_j | f_{a(j)})$$

With just one iteration, model 1 gives us equal probabilities for all possible alignments. Model 2 changes this. The reason Model 2 does not give us equal probabilities is because, unlike Model 1, it takes the positioning of the words in the sentence into account. So now the alignment probability distribution of the translation from the original language to its signed language, given that the first word is in position $i$ and the second in $j$, is given by $a(i, j, l_e, l_f)$. With this new information, Model 2’s word alignment becomes the following:

$$p(e, a|f) = \epsilon \prod_{j=1}^{l_e} t(e_j | f_{a(j)}) a(a(j)|i, l_e, l_f)$$

Finally, Model 3 takes into account when a word is not translatable from its source language to its respective sign language, by assigning it a tag with a **NULL** value instead, and the probability of it happening becomes:
Besides the IBM Model’s, to improve the results, they also use a string matching algorithm: more specifically, the Jaro-Winkler distance algorithm. It is given by:

\[ d_w = d_j + (l \times p \times (1 - d_j)) \]

The standard value for \( p \) is 0.1, and \( l \) represents the amount of characters that in the begging of the string are the same between S1 and S2, with a maximum of 4. Finally, the Jaro-Winkler distance is actually a specific version of the Jaro distance, and \( d_j \) represents it, between two given strings. It is given by:

\[ d_j = \frac{1}{3} \left( \frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m - t}{m} \right) \]

where \( m \) is the total number of equal characters between the two strings, and \( t \) stands for the amount of transpositions.

Indian researchers have also tried a translation method for Indian Sign Language (ISL) recently [1]. Inspired in previous statistical translation works [25], they have gone for a word based statistical method of translation, and like many other works before, they limited their corpora domain to something specific (in this case it was the indian railways), and was created with MOSES [26]. Something different about this work is that while others translate a certain language to its corresponding sign language (for instance, Vietnamese to VSL), this one translates English to ISL. This means that when they process the source text, they deal with problems at the level of the english level, and not indian, most likely to facilitate their jobs at doing so. The source text is tokenized and each token is changed for its equivalent gloss, found in their corpora, via a word to word direct exchange. Then, just like Othman and Jemni [25], they apply IBM models 1, 2 and 3. In order to evaluate their work, they first trained their corpus with an \( n \)-gram model (with unigrams, bigrams and trigrams). In Othman and Jemni’s work, they state that the resulting output can be pictured in the form of a matrix. They do this here, as shown in Figure 3.11.

Another recent project is called Dicta-Sign [27]. Dicta-Sign appears in 2010 and provides three state-of-the-art functionalities:

- a dictionary that allows to search through signs;
- translation between two signed languages;
- a Wiki for sign language.

It is available for British Sign Language (BSL), Langue des Signes Française/French Sign Language (LSF), Greek Sign Language (GSL) and Deutsche Gebärdensprache/German Sign Language (DGS). The second item in the previous list cannot be stressed enough: having a multi-lingual corpus allows many different scenarios, like translating one sign language to another. When a sign is not available, it
returns the closest-to-it signs (closest-match). Dicta-Sign uses SiGML to represent the signs.

### 3.2.2 LGP System

Even though most work in the area has been achieved in other languages, there has been some work done in European Portuguese [2]. This project focuses on the translation of Portuguese European to LGP, and the used architecture can be seen in Figure 3.12.
As observed in Figure 3.12, a gloss system was used. This project represents a good example of what was said in Section 2.2.2: when there is no corpus available, statistical MT stops being an option. This was the case here, so a rule-based system was made, with custom-made rules. A stemmer was used in the processing of the Portuguese text, as it allows us to see what a word’s stem and affixes are. The stem, in Portuguese, is associated with the word’s radical. The affixes (prefixes and suffixes) give us the option of looking at a word’s attributes, for instance, its gender.

As stated before, a POS tagger is also used for the syntax, and a Named Entity Recognizer used for gestural names. In LGP, to sign the name of a person, if that person has a known gestural name, it is used. If not, it is spelled, letter by letter. Through a bilingual dictionary, the words are exchanged for their glosses and the syntax changed from Portuguese to LGP. Then, for each gloss, a correspondent action is looked up from a database, implemented with JSON. An action may be made of one or more signs, as long as it fully represents its gloss. For these problems on Natural Language processing, NLTK was used. It is also interesting to note that the concept of hints was used as a tag that says whether the sign should be normal or spelled (like with people’s names), or if it is a numeral. Figure 3.13 shows an example of a sign’s contents. The id is relative to the animation. Then, we have the gloss and its list of actions (necessary actions to perform the gloss in its entirety). There is also room for contextualization through NLTK.

![Figure 3.13: Example of struct used in [2].](image)

57 basic signs were processed to use in the actions. This project, however, only considers sentences with a structure of a noun, a verb and then another noun because, like mentioned in Section 2.1.2, as far as it is known, LGP has a structure of Object-Subject-Verb (OSV). This is a limitation. Another downfall lies in the fact that there were no facial animations. Since plenty of important grammar aspects of LGP require non-manual elements, this is something else that needs to be done in order to get better results.

3https://www.json.org/
3.3 Discussion

The purpose of this section is to discuss the Implemented Systems in Section 3.2, and how they can be (or not) of help for our (or another) text-to-sign translation system.

In the first presented system – TEAM [3] –, the most interesting aspect it shows is the direct integration of its adverbs into the avatar, as shown with the example in Fig. 3.5. In this dissertation, this is not achieved as we are not using our own avatar. This is still an interesting approach to use in future work where the avatar used is of our own creation.

Grieve-Smith and Kanis et al. [21, 22] limit their projects to a closed domain: the first uses weather reports’ data and the latter uses train timetable information. Sometimes, it is better to work on smaller domains and build up from these, even more so in statistical machine translation systems.

The ViSiCAST project [23] uses GML to store its HamNoSys. In this dissertation, we do not use GML as we use SiGML instead (Section 4.1.1). The difference lies in the terminology used only. The same information is stored: as such, the choice of what XML language to use comes down to preference.

The vietnamese approach to this problem [16] inspired the syntax exchange in this dissertation. By using a rule-based approach for the word alignment problem, we can annotate syntax structures into our system (alongside their equivalent in sign language syntax). Anytime the system gets an input, it checks the existing syntax structures in the system.

Othman et al. [25] and Mishra et al. [1] are projects that help us better understand how a statistical approach in word alignment works, while Dicta-Sign [27] brings us an additional feature over the other systems, which is the possibility of also having a Sign-to-Sign translation, due to having multiple annotated sign languages in the system. For our portuguese language and LGP alone, this is not feasible: however, it is possible to have multiple variations of LGP in our system: anyone can choose (or annotate) their vocabulary in it, and anyone can choose (or annotate) their syntax rules.

Finally, Almeida et al.’s project [2] focuses directly on Portuguese and LGP. It lacks on syntax rules, as its only syntax change is dealt with the simplest form: Object – Subject – Verb. This dissertation goes further and adds syntax rules in order to make the system accept more syntax structures.
PE2LGP 3.0

Contents

4.1 Architecture ........................................... 28
4.2 Syntax Annotation ........................................... 43
4.3 Vocabulary Annotation ................................. 45
4.1 Architecture

The main goal of this project is to create an automatic translation system from European Portuguese to LGP that allows not only the translation of text input but also the annotation of syntax rules and vocabulary by anyone, trying to maintain simplicity in its process. In order to do this, we start by choosing a suitable Sign Language Intermediate Representation – Section 4.1.1 (defined earlier in this Thesis). The order of the system is the following: text is received as input (in Portuguese), and the output is the sign version ready to be processed by a 3D animated avatar. In between, there are two main modules: the **Sign Tokenization Module** (Section 4.1.2) and the **SiGML Generation Module** (Section 4.1.3). Each serves its separate purpose. The full model can be visualized in Figure 4.1.

![Figure 4.1: Visualization of the project’s architecture, from input to output.](image)

Before reading about each module, we will see what Transcription Language (Section 3.1) was chosen for our system (PE2LGP 3.0), and why.

4.1.1 Used Transcription Language

AZee [18] is the best option when it comes to functionality, because it lets us reproduce non-manual elements in time, regardless of the other manual elements. Unfortunately, after having contacted Filhol (AZee’s author), we received the answer that AZee is only yet a theoretical concept. On the bright side, as of now, a 3D movement translation using AZee is being worked on by Filhol, DFKI in Saarbrücken [1] and DePaul University in Chicago [2]. Until then, unfortunately, using AZee for this purpose does not seem possible.

Despite not being completely multimodal, SiGML using HamNoSys is still a viable solution. It is an XML variation, and as such, multi-modality can be easily added in the future. As it is out of the scope of this thesis to develop multimodality, it will have to be future work (in Section 3.1 the concept of multimodality is explained). eSIGN [3] is a software that allows for the creation of signs in SiGML, from HamNoSys (Figure 4.2). It is practical for the automatic generation of XML-like signs. It allows for the

---

1[https://www.dfki.de/web/kontakt/dfki-saarbruecken](https://www.dfki.de/web/kontakt/dfki-saarbruecken)
2[https://www.depaul.edu/Pages/default.aspx](https://www.depaul.edu/Pages/default.aspx)
3[http://vh.cmp.uea.ac.uk/index.php/SiGML_Tools#eSIGN_Editor](http://vh.cmp.uea.ac.uk/index.php/SiGML_Tools#eSIGN_Editor)
fingerspelling of words and typing numerals. However, its search by gloss feature is only available for English words. But with it, we can still manually annotate words for the dictionary used by our system. The fact that we can annotate words (signs) in eSign was a key factor in the choice of using HamNoSys for our system.

![eSIGN's window.](image)

Figure 4.2: eSIGN’s window.

SiGML is an XML variation. As such, it is not only easy to use, but also to modify. This is handful if we want to add a timestamp to it in the future (Section 6.2). For all of these reasons, SiGML (using HamNoSys) is the used transcription language in this thesis.

4.1.2 Sign Tokenization Module

4.1.2.A Overview

Figure 4.3 shows us how the input passed to the system (the sentence(s) to be translated) is processed from start to finish of this part of the thesis. In order to change the syntax, there is a morphology extraction step first, which includes POS Tagging, and a dependency parsing step in which we gather the syntactic dependencies of the input. With this information, the syntax is transformed into LGP’s. At the end, the annotated classifiers and non-manual expressions are also included into our Sign Tokens.

Before explaining each step in detail, we will briefly explain how the module works with an example: *O João come a sopa.* (in English: *John eats the soup.*).

The input, given as a STRING, can contain one or more sentences. The input is tokenized into Sentence Tokens (each Sentence Token is a sentence), and the Sentence Tokens are tokenized into regular tokens (elements of the sentence: words, named entities and punctuation). To perform this we use NLTK. To exemplify this step, let us suppose that our system receives as input the following STRING: *O João come a sopa. A Maria come a maçã.* (in English: *John eats the soup. Maria eats the apple.*). In
order to process each sentence on its own, we first separate them into *O João come a sopa.* and *A Maria come a maçã.* Then we tokenize each of the sentences.

Two main things are done for each sentence from the input: gathering necessary morphology information and exchanging the syntax from European Portuguese’s to LGP’s. The tool being used for these steps is Spacy 4, since it offers trained dependency models. Spacy allows us, for example, to infer that the *Sopa* (in english: *Soup*) is Singular (not Plural). Another important example is getting the verb tenses (and other information related to them). *Come* (from the example sentence *O João come a sopa.*) presents itself on the *Presente do Indicativo, 3ª Pessoa do Singular* (Singular Third-Person, Present Simple, Indicative mood).

Changing the syntax is done with the help of an annotated csv document, in which one of the columns contains possible syntactic structures in Portuguese, and in the other column, their equivalent syntax in LGP. Each pair is represented as a syntax rule. The rule that is applied to our example (*O João come a sopa.*) can be found in Listing 4.1. If a syntax rule is not paired with the input sentence, the original Portuguese syntax is used.

### Listing 4.1

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsbj PREDP-I obj</td>
<td>&lt;- nsubj obj PREDP-I</td>
</tr>
</tbody>
</table>

The rule shown in Listing 4.1, related to *O João come a sopa.*, does not account for the articles *O* and *a*. As an example, instead of having a rule for *art nsubj PREDP-I art obj*, we simply remove the articles and directly apply the sentence to follow the rule in Listing 4.1. This is because there is an

---

4https://spacy.io/
exclusion of words that are not used in LGP. The articles in the earlier example are not signed in its translation, and the rule is applied for the sub-sentence João come sopa. Its output becomes João sopa come.

Listing 4.1 says that when a sentence has a subject (nsubj), a predicate in the present tense of the Indicative mood (PREDP-I) and a direct object (obj), all of these in this specific order, the syntax is changed to subject, object and then the verb. The punctuation, in this specific rule, was also dropped: this happens because in a declarative sentence in LGP there is no need to sign a period. It appears in the input because we parse all of the syntactic elements of a sentence (including punctuation), and in the output, in this case, it disappears because it is not needed. As such, in the left-hand side of Listing 4.1 there is a period at the end of the syntax structure, while at the right-hand side there is not. With the help of such rules, the syntax is exchanged.

At the end, the Sign Tokens are returned in the form of a .txt file, in between brackets. A representation of the sign tokens for O João come a sopa. can be seen in Figure 4.4.

4.1.2.B Spacy’s Basics

In order to change the syntax from European Portuguese’s to LGP’s we need to determine the input sentences’ syntax. We also need morphological information about the words, for the signs to be signed correctly.

Spacy needs a language model to be used and we used the Portuguese model. Spacy works with a type of variable called doc. A doc variable receives as input a sentence, and in its output returns a list of its tokens, in accordance to the chosen model. This list of tokens contains information such as parts of speech, syntax dependency tags, lemmas, word shapes, stopwords or named entities (Figure 4.5). With this information, we can infer the sentences’ syntax and morphology.

As mentioned in the previous section (Section 4.1.2.A), we remove articles from our syntax rules. The doc variable that is created keeps information relative to the full sentence. This means that when omitting these words, no original information is lost. Only after storing the full sentence we omit the articles from the rules, since they will not be part of the resulting translation into LGP.
4.1.2.C Extracting Morphological Information

Knowing morphological information on our sentence will allow us to more accurately sign what we need, if we know beforehand that a certain token is a verb, adverb, a noun, etc. This type of information is called a token’s POS. Table 4.1 covers some of the POS tags returned by Spacy.

In order to sign the words in our input, additional information is needed, such as the word’s gender or number. If we look at the example of the word Cadela (female dog), in order to sign it we must sign Fêmea + Cão, which means that we need to know the gender of the word in order to sign Fêmea when needed.

Spacy provides a tag map for the Portuguese model. This tag map is an annotated file in Spacy’s repository, that contains this additional information (additional information besides the POS tags). Spacy does not allow us to access this information directly via already implemented code (as of writing this).
We can however read it, parse it and gather this information ourselves. It shows us, for instance, the main verb of a sentence. This is important because it immediately tells us the predicate of a sentence, as shown in Fig. 4.6.

"<cj7>|<mv>|V|PR|3S|IND|@FS-QUE"

Fig. 4.6 also shows us the pattern of these annotations. The tags are separated with a | character. Some (on the left) are in between < and > characters (like the <mv> tag) and others on the right are not. The final tag begins with a @ character. The first two types of tags indicate morphology information, while the tags that begin with an @ character show us syntax dependencies (which we do not need to parse from this document, since Spacy lets us get the dependencies directly through code). In order to parse the information in between the < and > we use a regular expression to get all the matches, while for the rest of it we use simple Python list operations.
Each first element of the tags that are not between < and > represents the main morphological class of the word: in other words, its POS (visible in Fig. 4.7). All of the possible POS tags that can be found in the Tag Map are shown in Table C.1.

"<COMP>|ADJ|M|S|@SC>" : {POS : ADJ},
"<E>|<sam->|PRP|@<ADVL" : {POS : ADP},
"<Eg>|<COMP>|ADJ|M|P|@N" : {POS : ADJ},
"<Eg>|<dem>|DET|F|S|@P<" : {POS : PRON},

Figure 4.7: A snapshot of a few lines that show full tags. The first morphological tag of each line contains information about the word’s main morphological class (in this case, ADJ, PRP, ADJ and DET, respectively).

In Fig. 4.7, after ADJ there is still M | S or M | P, while after PRP we have no additional tags. These represent (if they exist) additional information about the morphology of the word. In this tag map, different morphology class tags (POS tags) assume different patterns in regard to the remaining tags (to the right), which can be found in Appendix C.1.

Let us follow an example for better understanding. ADJ has a gender and a number associated with it. From the patterns in Appendix C.1, we can tell that having a gender and a number happens in the majority of cases, making it a default scenario. Then, we have morphological classes that have no additional morphology tags (for example: IN []). Finally, V (verbs) and PERS (personal pronouns) correspond to special cases.

Verbs, as shown in the patterns, have a verb tense, a person and number and a mood. These correspond to Portuguese grammar’s verb tenses and modes. Fig. 4.6, which presented us the tag <cjt>|<mv>|V|PR|3S|IND, is an example of this: the tag V tells us that the corresponding token is a Verb. Then, PR means it is on the Present verb tense, 3S means Singular Third person and lastly IND means Indicative mode (Indicativo, in Portuguese).

On the other hand, personal pronouns (PERS) have a gender, a person and number, and a case. Now we need to know what values these tags may hold and that we can use:

- **Gender**: M for Male, F for Female and M/F for both;
- **Number**: S for Singular, P for Plural and S/P for both;
- **Person & Number**: 1S, 1P, 2S, 2P, 3S and 3P are the possible values here. 1 denotes the first person speech, 2 the second and 3 the third. S and P denote the number (i.e: Singular and Plural);
• **Tense**: PR, PS, IMPF, FUT, INF, PCP, GER, COND and MQP. There can also be a PS/MQP. These mean, respectively: Present (Presente), Past Simple (Pretérito Perfeito), Imperfect (Pretérito Imperfeito), Future (Futuro), Infinitive (Infinitivo), Past Participle (Participio Passado), Gerund (Gerúndio), Conditional (Condicional) and Pluperfect (Pretérito Mais-Que-Perfeito);

• **Mode**: IND and SUBJ, which respectively mean Indicative and Subjunctive.

The gender and number add, as said before, additional information necessary in order to correctly sign female and plural words. The tense and mode of the verbs is also very important because in order to create syntax change rules, we need to associate rules with specific verb tense-mode pairs. This is because LGP **syntax changes according to the verb tense and mode being used in a specific sentence**. As an example, a sentence signed in the Present and Past of the Indicative mood will have a different syntax, with different Non-Manual Expressions. We can also extract the case, which can be the nominative case, the accusative case, the prepositional case and the dative case: these do not apply to portuguese and for now we will not be using them.

We can extract even more from the tags that are between < and >. This part is specially important in order to extract the sentence’s predicate (<mv> in Fig. 4.6). Some of the tags that we can use and extract information from are explained in the next list:

• **<mv>**: Main verb of the phrase, shown earlier within this Section as an example;

• **<date>**: A date;

• **<quant>**: Quantifier (muito, for instance);

• **<year>**: Any numerical year.

Not all of the tags that Spacy delivers are important in our context (see Appendix C.2 for more tags). These tags may have interest in, for instance, processing the various types of numbers (ordinal, cardinal, dates), but its main purpose for now is to identify the predicate. Spacy does not specify anywhere (to the extent of our search) the used terminology for these tags. We were able to find that these are presented in a mix of English and Portuguese VISL [28] notation tags. Some of the tags do not show up in any of their glossaries.

---

5 [https://visl.sdu.dk/tagset_cg_general.html](https://visl.sdu.dk/tagset_cg_general.html)

Table 4.2: Table depicting Spacy’s dependency tags and their equivalents in Portuguese Syntax.

<table>
<thead>
<tr>
<th>Dependency Parser Tag</th>
<th>Analogous Portuguese Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsubj</td>
<td>Subject [Sujeito]</td>
</tr>
<tr>
<td>ROOT</td>
<td>Root [Raiz da Estrutura Sintática (Em casos simples, o verbo)]</td>
</tr>
<tr>
<td>obj</td>
<td>Direct Object [Complemento Direito]</td>
</tr>
<tr>
<td>iobj</td>
<td>Indirect Object [Complemento Indirecto]</td>
</tr>
<tr>
<td>advmod</td>
<td>Adverbial Modifier [Modificador Adverbia]</td>
</tr>
<tr>
<td>nmod</td>
<td>Nominal Modifier [Modificador Nominal]</td>
</tr>
<tr>
<td>aux</td>
<td>Auxiliary [Verbo Auxiliar]</td>
</tr>
<tr>
<td>punct</td>
<td>Punctuation [Pontuação]</td>
</tr>
</tbody>
</table>

4.1.2.D Syntax Dependency Tags

Table 4.2 shows some of Spacy’s syntax dependency tags when used with the Portuguese model. Take our most used example sentence: O João come a sopa. Figure 4.8 shows its syntax being extracted from it, and its result. Through these tags we get the original Portuguese syntax from the input/sentence (or sentences), keeping in mind that Spacy is prone to mistakes while tagging the words. We are also only taking into account determined subjects, meaning that we only process determined subjects in this thesis.

![Figure 4.8: First, the sentence O João come a sopa. is shown, and then, its corresponding syntax dependency tags (as provided by Spacy).](image)

Processing the predicate with the help of these tags is tricky. Most verbs’ dependency tags in simple sentences are returned as ROOT by Spacy, and this happens because the tags are represented in a tree structure, and in these cases, the verb is at the root of it, being connected to all elements: the subject, the objects, etc. This is not always the case, however, for more complex situations. This means that we cannot rely on simply checking if a given token has the tag ROOT to assess if it is a predicate or not. In this project, we check if it is a VERB (Section 4.1.2.C) and also if one of its leafs (within the tree structure) corresponds to the subject. For a better understanding of this, see Figure 4.9.

Either a ROOT tag or some other tag, the tag that is delivered by Spacy as being the main verb is always incomplete: it does not give us the verb’s tense and mood. It is important that we use a dependency tag that does this, because LGP syntax changes according to each different verb’s tense and

mood pair. For example, if we use the rule input: nsbj ROOT obj. (relative to João come sopa.), we cannot tell what rule to use just by the tag ROOT. ROOT does not specify if the verb come is in the Present, Past or Future tense (or other). In the English language model, for instance, this is possible: by using token.tag, a tag that gives the verb’s information is given. As an example, if we load an English linguistic model in Spacy, and create a doc variable with the sentence John ate the soup, ate.tag is returned with a VBD tag (the VBD tag is used to indicate that the verb is in the past tense). As for the Portuguese language model, the tag that is returned is chosen from Spacy’s tag map (Section 4.1.2.C). Given this problem, using our last section’s gathered morphological information about the verb, we create our own tags for each type of verb. The list of tags that we change the verb to is presented in Table 4.3. Besides these, we also add -I or -S to the tags that start with PRED, to identify the mood: Indicative or Subjunctive. The tags end up being, for instance, PREDPP-I or PREDP-S. Please keep in mind that these tags were created by us, and do not have anything to do with Spacy.

Following the earlier example, we now may have rules for: nsbj PREDP-I obj., nsbj PREDPP-I obj. and nsbj PREDF-I obj.. As such, the verb come's tag in João come sopa. will be changed to PREDP-I, since the morphology extraction step tells us that the verb is in the Present Simple, Indicative mood. This way, this specific example will now differentiate between different verb tenses and moods, and follow its specific syntax exchange rule, leading to a different output.

4.1.2.E Changing the syntax to LGP syntax

We now have the original Portuguese syntax of the input given to the system, retrieved with Spacy. The syntactic transformation is now to be applied, according to Section 4.2. We created 100 sentences in Portuguese in order to accommodate 100 different syntactic phrase structures. Our goal was to hand these sentences to an LGP professional and have that person translate its syntaxes to the LGP equivalents (Figure 4.10). Not only was the syntax exchange annotated, but classifiers and non-manual expressions...
Table 4.3: Table depicting the verb tags that were created to associate to the syntax exchange rules.

<table>
<thead>
<tr>
<th>Verb Tag</th>
<th>Morphology</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREDP</td>
<td>Present Simple, Indicative [Presente do Indicativo]</td>
</tr>
<tr>
<td>PREDPP</td>
<td>Past Simple, Indicative [Pretérito Perfeito do Indicativo]</td>
</tr>
<tr>
<td>PREDPI</td>
<td>Past Imperfect, Indicative [Pretérito Imperfeito do Indicativo]</td>
</tr>
<tr>
<td>PREDF</td>
<td>Future Simple, Indicative [Futuro do Indicativo]</td>
</tr>
<tr>
<td>PREDMQP</td>
<td>Pluperfect [Pretérito Mais-Que-Perfeito]</td>
</tr>
<tr>
<td>INF</td>
<td>Infinitive [Infinitivo]</td>
</tr>
<tr>
<td>GER</td>
<td>Gerund [Geründio]</td>
</tr>
<tr>
<td>COND</td>
<td>Conditional [Condicional]</td>
</tr>
<tr>
<td>PCP</td>
<td>Past Participle [Participio Passado]</td>
</tr>
</tbody>
</table>

Table 4.3: Table depicting the verb tags that were created to associate to the syntax exchange rules.

that are required to sign the words were included.

From these annotations, manual rules were created for each syntactic phrase structure, where for a received phrase structure, the output would be the resulting respective annotation. These annotations were normalized in order to make the input and the output have the same notation and simply rearranging its order (for instance, \( A B C \rightarrow A C B \)). The used notation was Spacy's dependency tags directly, alongside the predicate variation added by us in the previous section. For the sentence O João come a sopa., it would translate to \texttt{nsubj PREDP-I obj .} , and then, the output would be (according to our annotations) \texttt{nsubj obj PREDP-I}. The rule is applied and the sentence becomes \texttt{João sopa come}. A list of some rules can be found below:

\[
\begin{align*}
\text{nsubj PREDP-I obj .} & \rightarrow \text{nsubj obj PREDP-I} \\
\text{nsubj PREDP-I obj ?} & \rightarrow \text{nsubj obj PREDP-I ?} \\
\text{nsubj aux INF obj ?} & \rightarrow \text{nsubj obj PREDF-I ?} \\
\text{nsubj INF obj ?} & \rightarrow \text{obj INF nsubj ?}
\end{align*}
\]

Having explained the process, there were a few cases in which the sentences and their rules had to be dropped because the annotation that was given to us would sometimes include different words from the ones that were given as the input. As an example, let us look at the word \texttt{deliciosíssimo}, which was translated as the signs \texttt{doce} and \texttt{bom}. One can quickly realize that it is not trivial to go from one word to the other set of words, except by making exceptions for those cases. It is not at all ideal, but for now, we do not see any other way to deal with this problem. These exceptions are introduced in the next Section.
After this step of manual rules that change the syntax from Portuguese’s to LGP’s, because we need to incorporate the classifiers and non-manual expressions into these rules, we perform a second set of rules to take care of this. These rules take as input the LGP syntax of the sentence, and output the same structure, but with the classifiers and non-manual expressions involved (Listing 4.2). A notation was created for this and it can be defined by:

- [ ] square brackets are used for classifiers (example: \texttt{nsubj obj PREDP-I[obj]}. By following this rule, we are not just adding a verb to that position: we are adding the verbal classifier between the verb and the object of the sentence. For João sopa come, the result would be João sopa come[sopa], and in the next module, when fetching the sign for come[sopa], the system will know we are dealing with a verbal classifier);
- {} curly brackets are used for non-manual expressions (example: \texttt{nsubj obj PREDP-I[objc][?]}. The character ? adds an interrogative non-manual expression, resultant from the input being a question).

A list of all of the Non-Manual Expressions that can be used can be found in Appendix E;

\begin{verbatim}
: nsubj obj PREDP-I -> nsubj obj PREDP-I[objc]
\end{verbatim}

\textbf{Listing 4.2:} Syntax rule that is applied to \texttt{O João come a sopa.}’s LGP syntax structure (more precisely, João sopa come). Afterwards, it becomes João sopa come[sopa]

At the end of these two rule-change steps, we have our syntax fully exchanged, and our \textbf{Sign Tokens} are \texttt{outputted} into a file with the format shown in Fig. 4.4 from Section 4.1.2.A.

\subsection{4.1.3 SiGML Generation Module}

\subsubsection{4.1.3.A Overview}

In Figure 4.11, we can take a look at how the second and final part of this thesis runs from the parsing of the \textbf{Sign Tokens} (generated from the first part of the thesis) to outputting the SiGML file required to play our translated animations.

By the end of the first half of this thesis, the \textbf{Sign Tokens} are kept in their own .txt file. This allows for the full separation of both parts. If, in the future, we want to improve one of these parts, we can do so without messing with the other one.

As a means of demonstration, let us follow the example of the sentence: \texttt{O João come a sopa.}. The input file is read, and the \textbf{Sign Tokens} are stored as seen in Listing 4.3: in a list of tuples where the first element is the token and the second element a dictionary of all of its attributes.
Figure 4.11: Overview of the second part of the thesis. The system receives the Sign Tokens and outputs a file with the SiGML necessary for the 3D animations.

Listing 4.3: Stored Sign Tokens’ format.

```json
1 {'Joao': {'gender': 'M', 'type': 'PROP', 'number': 'S', 'syntax': 'nsubj'}},
2 {'sopa': {'gender': 'F', 'type': 'N', 'number': 'S', 'syntax': 'obj'}},
3 {'come': {'person': '3', 'mode': 'IND', 'time': 'PR', 'syntax': 'PREDP-I',
4 'type': 'V', 'classifiers': 'obj', 'number': 'S'}
```

Afterwards, with the help of these attributes, a number of rules is applied to our example within the LGP Rule Exchange Submodule. The token João is exchanged for the tokens J, O, A and O, because it is a name of a person. Sopa and come are merged into a single token in order to form a classifier: sopa, come.

The next step is changing each token for their SiGML equivalent with the help of our dictionary. The words have been annotated using eSign, a program that transcribes HamNoSys into SiGML. Each tokens’ SiGML is fetched and put into a single SiGML file. The resulting .sigml file will contain our full SiGML, ready to be reproduced by a 3D avatar.

4.1.3.B LGP Rule Exchange Submodule

This module’s input is the Sign Token file generated at the end of the first module (Section 4.1.2). The Sign Tokens in their output file come in their own LGP syntax order, and bring their morphological information with them. The first step of this module is reading it and storing it in a python dictionary.

After storing this information, we still have to send each Sign Token under a set of rules. In the next step (after this Section), we exchange, token by token, the Sign Tokens for their SiGML equivalent. To do this, we look up a dictionary made of up SiGML files for each possible available token to exchange. This
means that in [João, sopa, come], we would look for the files João.sigml, sopa.sigml and come.sigml. However, João is a name of a person, and as such it needs to be fingerspelled, which means that we need to look for the letters J, O and A. Moreover, sopa and come make up a verbal classifier, as the combination of both tokens make up a special sign combination. This submodule, LGP Rule Exchange Submodule, grabs the Sign Tokens and exposes them to these set of rules, so that they can be properly exchanged for their correct SiGML equivalent. Following these rules, instead of looking for João.sigml, the system would now look for J.sigml, O.sigml, A.sigml and then O.sigml once again.

This submodule treats the Sign Tokens with the following rules:

- **Fingerspelling**: if the submodule finds a token that represents a name, it breaks it into letters. Example: [João] → [J, O, A, O];
- **Gender**: if a word’s gender is the female gender, the FEMALE sign is added before the word itself;
- **Classifiers**: classifiers are treated in this submodule, for the system to look the classifier up correctly.

### 4.1.3.C Dictionary Creation and Token Exchange

As said in the previous Section (Section 4.1.3.B), each Sign Token, by its order in LGP syntax, is exchanged directly for its SiGML equivalent in our SiGML dictionary. When the word is not yet available on the dictionary, it is fingerspelled: for this reason, the whole alphabet is annotated. However, there is an underlying process in the creation of our dictionary, which we were able to verify in Fig. 4.11, in Section 4.1.3.A. The same bit is presented below in Figure 4.12.

![Figure 4.12: Overview of the part of this module that deals with the creation of our SiGML dictionary.](image)

SiGML is the XML version of HamNoSys. This means that the content of a SiGML file is made up of HamNoSys characters, but in a simple format of XML. Our dictionary contains SiGML files of a limited set of words. When we get to the Dictionary Creation submodule, all we do is write an initial line with a SiGML opener tag (<sigml>), iterate over our dictionary and extract all of the needed SiGML (and write them on our output file) and at the end write a final line with a SiGML closer tag </sigml>). An example can be seen in Figure 4.13.
Figure 4.13: An example of a SiGML file. It has the opener and closer tags on the first and last line, and the letter J's equivalent in HamNoSys (in order to give a simple example).

In the earlier figure, we can still see that the tag `<hns_sign_gloss>` defines the limits of a certain gloss (or token). In this case, based on the example, it contains the information for the letter J. Inside it, it has two other tags: `<hamnosys_nonmanual>` and `<hamnosys_manual>`. These are self-explanatory: the first tag contains the HamNoSys information for the non-manual expressions of the signing of that gloss, and the latter contains the information for the manual parts of the signing of the same gloss. In the case of the letter J, it only contains manual expressions, which leaves the `<hamnosys_nonmanual>` tag empty. As for the tags inside of the `<hamnosys_manual>` tag, these correspond to HamNoSys characters translated from basic HamNoSys to their SiGML equivalent.

4.1.3.D Final Output

The final output is represented in a single .sigml file, with our sentence(s) represented in SiGML.
4.2 Syntax Annotation

We concluded that it would be helpful if we could have a corpus of normal Portuguese syntax examples as input and the equivalent syntax in LGP as output, so that we could generate rules that take care of the syntax transformation (example: Subject Verb Object → Subject Object Verb). As such, we have created 100 sentences in Portuguese, in an attempt to cover plenty of Portuguese’s syntactic functions. To fill in the output, we had an expert in LGP do the job in a specific format, respecting the order of the final sentence and adding things like classifiers to the result (Figure 4.14). This expert is an LGP native speaker with a bachelor’s degree in LGP, a Master’s Degree in Sign Language and Deaf Education and is also currently finishing a PhD in Cognition and Language.

Figure 4.14: A print-screen of some sentences, input and output. Just with three sentences, we study the difference between past-tense of the verb and interrogation. Note: ENM stands for Non-Manual Expression.

Appendix B shows the used sentences and their respective annotations. The sentences try to cover things like punctuation, negation, different types of sentences or different syntactic functions. The output not only changes the syntax, as initially intended, but it also reveals important grammatical behaviour.

The main goal of these annotations was to generate rules that were standardized for every situation possible of a similar syntactic translation. For example, if we annotate a rule that picks up on the sentence “O João come a sopa.”, we want to make sure that this rule is applied to all sentences that own, in order, a subject, a verb in the present simple and a direct object. In other words, we have the following rule: “nsubj PREDP-I obj. → nsubj obj PREDP-I” (terminology is discussed in Section 4.1.2 - for now, we just need to know that nsubj is the subject, PREDP-I is the verb in the Present tense of the Indicative mood and obj is the direct object). Every sentence that is passed as input and has that same syntax - nsubj PREDP-I obj. - will now be syntactically translated to the output of the same rule - nsubj obj PREDP-I (Fig. 4.15).

Fig. 4.15 only represents one of the rules that were generated: a very simple scenario. The more complex the sentences get, the more we are interested in knowing what those specific syntaxes’ outputs will be. However, many of the rules had to be discarded due to the inability of transforming the annotated outputs into a “syntax-only” terminology output. Let us take Fig. 4.16 as an example, which shows us the process of analyzing the output written by our LGP specialist, and with that infor-
Figure 4.15: The annotated sentence was the original sentence used to create our rule. Afterwards, we can apply this same rule to other sentences with the same syntax, just like Possible Input 1 and 2 in this Figure.

Annotation Sentence:  O João come a sopa.
Output:  João sopa come

Possible Input #1:  A Maria compra a comida.
Output:  Maria comida compra

Possible Input #2:  O Bobi morde a trela.
Output:  Bobi trela morde

Figure 4.16: We create a rule after looking at the annotation: PREDP-I[obj] represents the verbal classifier between that verb and object.

Now let us look at a different example, in Fig. 4.17. First, let us remind ourselves that we are simply dealing with syntax exchange (i.e: word alignment). Given Fig. 4.17's input - the sentence “Boa noite, amigos.” -, the only words we can possibly change are the ones in that sentence: BOA, NOITE and AMIGOS. But in the annotation given in that same example, another word is added: AREA. If this translation was to be done only for this sentence in specific, than this example would be usable: we would simply add the AREA to our final output. However, we are trying to generate a rule that acts as a mold for all of the sentences that may have that same syntactic structure, and as such, this becomes unusable. There are far more difficult situations of this matter that were found in our annotations and, as such, they have been discarded. Those sentences can also be seen in Appendix B: they are represented in greyed out rows.

Finally, there is a second set of sentences which were simply added as exceptions. If the input is any of those sentences, it just translates them directly. In other words, the sentence is not translated
from its syntactic structure but from the full sentence itself. These exceptions are for common greetings sentences, exemplified in Fig. 4.18. Exceptions can also be seen in Appendix B: they are represented in blue rows.

![Figure 4.17](image)

**Figure 4.17**: An example of an exception. As can be seen, the rule itself is only applied when a specific sentence is found on the input.

### 4.3 Vocabulary Annotation

In order to use a SiGML dictionary (as shown in Section 4.1.3.C), we need to generate SiGML files for the words that will be available in our dictionary (Fig. 4.19). This is where eSign comes in handy: it receives input in the form of HamNoSys and outputs its SiGML equivalent. With eSign, we can build our dictionary.

In order to pass HamNoSys as input, we need to know HamNoSys. The official documentation is easy to understand, and it does not come across as something that we need to fully know before trying to write in HamNoSys: instead, it comes across as a glossary that we can check on when trying to do certain things with our sign. These certain things are handshapes, orientations, locations and actions (movements). Every sign can be built with these things, in that same order, specified in

---

a HamNoSys line (Figure 4.20). The different types of handshapes, orientations, locations and actions can be checked on the official documentation.

The software eSign will take our HamNoSys input, but we need to know what we are signing and how to sign it. In our case, this is done with the help of the mobile application/website SpreadTheSign, which has plenty of annotated videos. With the help of these videos, we manually write their equivalent HamNoSys.

Let us follow the example of annotating the word sopá (in English, the word soup). We start by searching for the word sopá in SpreadTheSign's website and learning how it is signed (Figure 4.21).

In order to write sopá's equivalent in HamNoSys, we need to write its configurations, its orientations (palm orientation and hand orientation), locations and movements, all of these as stated earlier. If we write, for instance, only one configuration alone, it is going by default to be the dominant hand's configuration. If we want to write a configuration for both the dominant and non-dominant hand, we need to do so as shown in Figure 4.22.

---

The first thing we annotate is the configuration and orientation of each hand (Figure 4.23). Each side (dominant and non-dominant) has three characters: the first is the configuration of the hand, the second is the hand orientation and the third is the palm orientation. The hand orientation is defined by the direction in which we would be pointing at, if we were pointing at something with our hand. The palm orientation is the direction our palm is facing. If we take a look at SpreadTheSign’s annotated video of the word *sopa* (Fig. 4.21), we can see that both hands have the same palm orientation (palm facing upwards) and that the hand orientation in each of them is the opposite of the other (one points to the right, and the other to the left). Such is visible in the HamNoSys characters: the characters related to hand orientation show arrows pointing to each side, and the palm orientation characters both show a black shading pointing up (which means the palm is pointing upwards). As for the configuration characters, they both represent open hands, but the dominant hand (the one that moves) is bending a little, and also has its thumb popping out.

After the configuration (handshape) and orientation, we annotate the location and movement (Fig. 4.24). This time, for this specific sign (*sopa*), we only need to separate the locations for each hand. This is because, in this sign, only the location of each hand is different. As for the movement, only the dominant hand moves. By simply assigning a movement without hand separation, the movement is applied to
the dominant hand by default. The location of each hand corresponds to the initial location before the movement occurs. As such, one of the hands is located directly under the chin, and the other at chest level. The second character on each hand is the distance to the body - in this case, there is contact with their corresponding body part. To finish this annotation off, we annotate the movement: in this case, represented by an arrow down, an arrow up and a + sign. The arrows indicate that the dominant hand will perform a down and up movement. The + indicates one repetition, which means that after moving down and up, it will repeat that same movement, moving down and up again.

![Figure 4.24](image)

**Figure 4.24**: Partial HamNoSys line of the word *sopa* – location and movement. The hand location is divided for each hand (dominant and non-dominant), and only after comes the movement. Because the movement is not specified for both hands, it only applies for the dominant hand, which means that only the dominant hand will move.

In order to write the HamNoSys, we can use **eSign**. **eSign** provides us with a HamNoSys keyboard so that we can easily input what we want in it (Fig. 4.25).

In order to test our SiGML, we can use **JASigning**[^10]. **JASigning** is an avatar that reads SiGML and plays its animation in a 3D avatar. For simple animations that test our words, it works. It is represented in Fig. 4.26.

[^10]: http://vh.cmp.uea.ac.uk/index.php/JASigning
Figure 4.25: *eSign*'s HamNoSys keyboard available, which enables our HamNoSys writing.

Figure 4.26: Example of JASigning usage with the word *sopa*. 
Contents

5.1 Syntax Evaluation ........................................... 51
5.2 SiGML Annotation Evaluation .............................. 53
5.3 Signing Evaluation ........................................... 60
This section is dedicated to the evaluation of the thesis. The full evaluation is divided in three evaluations:

**Syntax Correctness (Section 5.1):** this evaluation checks how many sentences from given sets are matched with syntax exchange rules;

**SiGML Ease of Annotation (Section 5.2):** the second evaluation of this thesis gives people the task of annotating words with eSign, justifying its use. This evaluation is done in accordance to one of the goals of this dissertation, which is making sure that the annotation of the system can be done by anyone who deems necessary, trying to keep the process of learning how to do it simple;

**Final Signing’s Readibility in a 3D avatar (Section 5.3):** in this evaluation, experts are shown signing by an avatar (JaSigning) and evaluate it on readability and precision. The evaluation is done showing signs annotated by the sistem in the JaSigning avatar as a way to establish a bridge between what the system generates (the intermediate text representation) and what the experts are able to understand and read (the signs in the format of a 3D avatar). Since the system’s first vocabulary was annotated by us (non-experts) with the help of SpreadTheSign, this evaluation is important in order to conclude if such an annotation can be trustworthy.

### 5.1 Syntax Evaluation

#### 5.1.1 Experimental Setup

Syntax Evaluation consists on evaluating the probability of matching a sentence on the input with one of the syntax translations, given a new set of random sentences. Given that the syntactic translations occur through a finite set of rules (more could be added later on), we can predict that only if we give as input to the system a sentence that has the same syntax of a specified rule, will the system make a syntactic translation.

With this in mind, instead of creating just one set of sentences to be evaluated, we created two sets: one set purposely created in accordance with the specific rules of the system, and another set created not in accordance to any syntax rules (not having in mind the rules of the system). Both sets are made up of 50 sentences and can be found in Appendix D.1. The main goal of the first set is to verify why a syntax translation could go wrong, even if the rule for it exists.

It should be clear that when the system does not find a sentence with the same syntax as the one in the input, it translates the sentence with the original syntax and this will result in the translation having Portuguese syntax and not LGP syntax. When, on the other hand, the system finds a rule specific to
that sentence, that rule is triggered. Only this second case will be considered a success.

5.1.2 Results

Results are shown in Fig. 5.1.

![Figure 5.1: Number of sentences, for each set of sentences, that had a matching syntax rule applied to them.](image)

Before analyzing the random set of sentences, we will analyze the set of sentences that were created having in mind the existing rules. As we can see in Fig. 5.1, 30 out of 50 found a matching rule, while the other 20 did not. Because the sentences were made trying to emulate the same syntactic structure of the annotated sentences, it means that there are a few factors that are triggering this outcome. To analyze this outcome, we need to analyze these sentences.

The biggest problem in these sentences is the dependency tag that is assigned to each token by Spacy, when gathering a sentence’s syntax. Spacy uses a probabilistic model, which is not perfect, to assign the morphosyntactic category to a word: as such, in some sentences, even if we correctly apply the same syntactic structure, Spacy may still assign it differently. This is something that we have to live with. An example lies in the sentence *Vais vender a casa?* (Appendix D), to which Spacy assigns the following syntax: `nsubj INF obl ?` (a subject, a verb in the infinitive form and an oblique nominal). The rule that we were trying to trigger with this test sentence was the one of: `nsubj INF obj ?`, and its original annotated sentence was *Vais comer a sopa?*. Notice how we want *casa* in the example sentence to be identified as a direct object of the predicate (identified with the tag *obj*), but instead, it is identified with a different tag by spacy (it is assigned the tag *obl*). As such, the rule that we have in our system is not triggered. As it is a Spacy problem, the most that we can do in the future is to cover these cases and create rules for them as well.

Let us now look at the random set of sentences. It is no surprise that only 9 had a matching syntax. There is a total of 63 rules that can be matched with the input. Obviously, there are plenty more ordered syntax combinations that can be passed in the input and, as such, the probability that a sentence will fall under a rule that currently resides in our system is very low. Random sentences show us the biggest liability in a rule-system: we have to be able to dynamically cover more and more rules in order to accommodate more and more combinations of syntactic structures passed in the input. It is important to
note that the factor that depicts if a rule is triggered or not is if the rule exists or not: the more rules we add, the more this random set's results should improve. Rules can always be added in the future, which means that this system works, and can evolve through time.

5.2 SiGML Annotation Evaluation

5.2.1 Experimental Setup

The main goal of this evaluation is to understand how easy and practical it is to add new entries (new words) to our SiGML dictionary through eSign, for people who have never experienced HamNoSys and with little instruction on how to use it. To do this, we used Google Forms and delivered a test form with a few steps:

- **Personal Introduction**: here, the testee fills in his age, gender, affiliation and if the testee has previous knowledge on sign languages and HamNoSys;
- **Requirements**: instructions on how do download and install the required software in order to perform this test;
- **Introduction to sign languages**: step that briefly explains the concept of sign languages and its minimal pairs (handshapes, orientations, movements, locations and non-manual expressions, as explain in Section 2.1.1);
- **HamNoSys Introduction**: here, the basics of HamNoSys are given to the testee for him to read and learn. The HamNoSys documentation should be kept as a glossary for the next steps;
- **eSign Introduction**: this step provides an introduction to the usage of eSign;
- **Annotation**: here, the testee is asked to annotate 4 different words as signs into eSign, using HamNoSys. This is the main testing step of the whole evaluation, and also the step that takes the longest. The testee is asked to keep track of approximately how long it took him/her to finish each separate annotation;
- **Questionnaire**: in the Questionnaire step, they fill in a simple questionnaire that evaluates the easiness of the annotation through eSign.

The questions that are asked in the last step (Questionnaire) to the testee are shown in the next list:

1 - **Did you find the theoretical introduction about sign languages to be easily comprehensible?**
   Evaluated from 1 to 5 (Hard to comprehend – Easy to comprehend);

2 - **Considering that you had no previous knowledge of HamNoSys, nor of how it would be represented in a writing system for Sign Language, how easy did you think it was to learn**
3 - If you answered with the values 1, 2, 3 or 4 to the previous question, what would you say that was the most difficult part in the comprehension and learning how to use HamNoSys?
Open answer (text);

4 - Did you think using eSign was intuitive?
Evaluated from 1 to 5 (Little Intuitive – Very Intuitive);

5 - If you answered with the values 1, 2, 3 or 4 to the previous question, what did you think that was less intuitive about using eSign?
Open answer (text);

6 - How easy was the usage of HamNoSys’s keyboard in eSign, for the annotation of words?
Evaluated from 1 to 5 (Very difficult – Very easy);

7 - If you had to evaluate the precision of your annotations in comparison to SpreadTheSign’s videos, how would you evaluate them?
Evaluated from 1 to 5 (Not at all the same – Exactly the same);

8 - Last but not least, keeping in mind that this was a sign language word annotation task, do you think the annotations of the 4 words consumed too much time?
Evaluated from 1 to 5 (They hardly consumed any time – They consumed too much time);

This test was divided into 2 batches of 5 tests each, in which the first batch the SiGML was not gathered and the in the second one the SiGML was gathered as well. The first batch served the purpose of seeing if this test was feasible, and its answers can still be used.

This test was made on Google Forms; the first batch was sent directly as a link to the testees, and in the second batch the tests were made in a room with us guiding the person through it. Even though we were there, there was generally no guidance needed as the test itself (the form) explained everything in detail. To the testee, it was explained that he could always go back to the explanation pages in order to remember the information taught about Sign Languages, HamNoSys and eSign. The annotation teaching process was very similar to what was explained in Section 4.3. By trial and error, the testees would annotate a sign, test it on JaSigning, and re-annotate it according to what they thought was wrong in the previous iterations until they got an end result they were happy with.

5.2.2 Results

Each test lasted from 1h30m to around 2h. As stated before, there were two batches of 5 tests each. Most answers, however, can be analyzed from the 10 testees all-together.
First, from the *Personal Introduction* page, we get to know that 1 of the test subjects had his age in between 25 – 44, and the other 9 had their age in between 15 – 24. As for the sex, 7 were males and 3 were females. Then, for their education, 7 had a bachelor’s degree, while 3 had a master’s degree. Of all 10, 7 studied Computer Science (*Engenharia Informática*), one Physics and another Environmental Engineering. One testee only stated that he graduated in Engineering (not specified). None of the testees had previous knowledge on LGP. This was, from the start, a requirement: however, it was inserted as a question just to confirm this. Just like LGP, none of the testees had previous knowledge on HamNoSys.

Now we step into the *Questionnaire* page of the test. Let us label each question with the number associated to each of them in Section 5.2.1. We will start with Question 1. In Question 1 it was asked to the testees if the theoretical introduction to Sign Languages presented in the *Theoretical Introduction* page was easy to understand. The results are shown in Figure 5.2.

![Figure 5.2: Chart that presents the results to the Questionnaire’s first question.](image)

The *y-axis* represents the absolute frequency of each possible value in the *x-axis*. We can see that for Question 1, 3 people answered 3, 4 people answered 4 and 3 people answered 5. Given that these people had no previous knowledge of Sign Languages and that no 1’s or 2’s were answered, instructing the basics of Sign Language (sufficiently enough for people to be able to annotate words afterwards) to people is feasible.

Question 2 gives us more diverse results. It was asked to the testees if they thought that the learning of HamNoSys was easy, given that they had no background knowledge of it. The results are presented in Figure 5.3.
The fact that no one thought that it was very easy tells us that there was always some difficulty related to learning HamNoSys. Out of the 10 total people, 4 answered 1 or 2. Indeed, learning HamNoSys can be too much information at once: but this was just a quick experiment. These results still prove to be good given the fact that all of the testees had no previous knowledge on sign languages or HamNoSys in general. The next question asked the testees that if they had answered within the range of 1 – 4 on the HamNoSys, what had been the most difficult thing in comprehending HamNoSys. Most of these answers cannot be taken into consideration as some testees confused learning the language HamNoSys with using it in eSign afterwards. Of the ones that could be taken into account, one said that there were so many symbols available to use that it could be overwhelming. This is normal, as the testees are learning everything about sign languages for the first time. Other answer surprisingly stated that the testee did not expect to learn such a language in less than an hour. Even though this does not answer the question of what was the most difficult thing to learn about HamNoSys, it tells us that this testee in particular did not find anything more difficult in specific.

Question 4 asked the testees if they thought that the eSign software was intuitive to use. The results can be seen in Figure 5.4.

Out of all of the testees, 6 answered very intuitive. Two testees answered 3, one answered 4 and the other answered 2. The next question asked the testees to specify what they found the least intuitive about eSign in case they answered within the range from 1 – 4, and as such there were only 4 answers to this. All of these answers stated the same thing: the panels for new utterances and signs were not
very intuitive. This is an eSign's interface problem. Only signs were being generated in eSign, and not utterances. However, in order to create a new sign, it must be done inside at least an empty utterance. This is the part that confused 4 testees. With a little previous knowledge on this, or even by getting to know the software, this does not pose a big problem.

Question 6 was related to the ease of use of HamNoSys's keyboard in eSign, and the answers can be checked in Figure 5.5.

The keyboard was found to be very intuitive by 6 testees. Only one answered with the number 2.
The final two questions (Question 7 and 8) are the most interesting ones. Question 7 asks the testees to evaluate their final avatar animations compared to SpreadTheSign’s. Question 8 asks the testees if it took them a long time to annotate the words, given the fact that this was a sign language annotation. The results to both can be found in Figure 5.6.

![Figure 5.6: Chart that presents the results to the Questionnaire’s seventh (on the left) and eighth question (on the right).](image)

The majority of testees thought the annotations took a long time: 4 testees answered 4 to Question 8. However, the answers distributed in a way that not many conclusions can be drawn. The only way to draw conclusions about the time it took to annotate words is to analyze the time that it took for each testee to annotate each word, which was asked for the testee to submit in the previous page (the Annotation Page of the evaluation form).

On Question 7, the results tell us that 4 testees answered 4. Even though the numbers are evenly distributed through all values (except for 4), having half of the testees answering 4 or above has an additional weight to it. Because everything is new to them (sign languages, HamNoSys and eSign), it takes plenty of confidence to state that their animations look very alike the ones in SpreadTheSign: this means that they knew what to do and were able to achieve the end goal. In order to try and confirm this, we can analyze the SiGML that each testee used and see if it matches what they answered here. We can only analyze 5 results, because only the second batch of tests asked the testees to submit their SiGML into our evaluation.

As such, to complement Questions 7 and 8 we will now analyse the Annotation Page.

After analyzing each person’s SiGML for each word (the words were RAPAZ, RAPARIGA, CARNE and BOLO), it was interesting to see that only two answered 4 to Question 7. One of the 5 people also delivered invalid SiGML, so we will discard his test. Of the remaining 4, the worst SiGML was from one of the testees who answered 4 to Question 7. The rest were very similar, and close to the end result, even if the testees thought they had a poor performance on it. By analyzing the SiGML further, it is
easily concluded that there was one main difficulty while annotating and testing the signs for the first time: the orientation. By looking at the SiGML’s and playing them on JaSigning, the hand configurations and movements all look alike, and are correct. However, most hand or palm orientations are slightly off. This may be due to the fact that when a HamNoSys symbol points right, for instance, it is pointing to the right side of the signing avatar (in the signing avatar’s perspective, and not in our perspective). When learning HamNoSys, choosing the right orientations got confusing.

Figure 5.7 shows us the mean and standard deviation of the time it took for each testee to annotate each word. The results show us that the hardest annotation was the first one. From then, there is a decline in the time taken to annotate each word. The only exception is the second and third word, which took about the same time. Still, a learning curve can be seen: as the annotations progress, the easier it gets, and the more accustomed the testee is with annotating.

![Time taken (mean) by each testee to annotate each word](image)

Figure 5.7: Time it took for each person to annotate each word.

From all of these results, it is possible to say that if these results were gathered with people who have never had contact with sign languages, much better results should come from someone who is a specialist. A feature of this system is that anyone can annotate it. For now, there is not much vocabulary inserted in the system, but if someone wants to add more, little instruction should suffice. Besides, even if a sign is wrongly annotated into the system, it can be reported by someone who notices and re-annotated.
5.3 Signing Evaluation

5.3.1 Experimental Setup

In this final evaluation, we have specialists looking at a set of annotated signs and evaluate them. In order to do this, we show them these being signed by an avatar: JaSigning. JaSigning is an avatar that reads SiGML and plays the signs in it. It is not meant as a final avatar software for this system in particular, but for testing this part of the system. An example of how it works can be seen in Fig. 5.8.

![Image of avatar and code snippet]

Figure 5.8: Snapshot of a SiGML avatar and the input from where it is reading.

In order to perform this test, just like in the previous evaluation (Section 5.2), Google Forms was used. In order to complete the test, the testees are first given a Personal Introduction form to fill, like the one in Section 5.2.1 (they give information such as their age and affiliation). Afterwards, a quick tutorial of how the test will work is presented to the testee, with an example word: not only is that word shown in video format of the avatar signing it, but also its SiGML is provided in case the testee wants to test it for themselves in JaSigning. Then, 5 words and 5 sentences are presented in the form of avatar movements with JaSigning, the same way in the previous example. For each of the words/sentences, the testees are asked to write what they were able to perceive from the avatar movements (i.e.: what was the word/sentence that the avatar was signing). Finally, they evaluate these on readability and precision, and if it was not readable or precise, they write what was wrong with the avatar's sign.

Readability is related to the avatar movements and signing itself, and how natural it looks/how
perceivable its movements are;

**Precision** deals with the linguistic correctness of the word/sentence itself, from syntax to a sign being wrong, etc.

Both of these concepts are explained explicitly in the test. The words and sentences chosen were picked at random, and were annotated using eSign, through HamNoSys, using the process shown in Section 4.3. Below, the list of words and sentences is presented:

<table>
<thead>
<tr>
<th>Words:</th>
<th>Sentences:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(W1) Cristina</td>
<td>(S1) O Tiago come a sopa.</td>
</tr>
<tr>
<td>(W2) Lisboa</td>
<td>(S2) Boa tarde.</td>
</tr>
<tr>
<td>(W3) Carro</td>
<td>(S3) O Ricardo não escreveu.</td>
</tr>
<tr>
<td>(W4) Mapa</td>
<td>(S4) A Rita conduzirá a mota.</td>
</tr>
<tr>
<td>(W5) Ir (verbo ir)</td>
<td>(S5) Tu treinas o cão?</td>
</tr>
</tbody>
</table>

### 5.3.2 Results

There was a total of 3 testees for this evaluation. All of the testees were between the ages of 45 and 64 years old, and their sex was female. One had a bachelor’s degree, another a master’s and the other a doctorate, which ranged from Linguistics to Psychology. On the question: *On what level do you consider your LGP to be?*, which could be answered from 1 to 5 (Basic – Native Language), 2 of the testees chose 3 and the other testee chose 5.

Figure 5.9 shows the overall results for each word’s and sentence’s precision and readability. What we can tell right away is that for each word and sentence is that the chosen values for both the precision and readability were the same (except in the first word). The testees did not seem to differentiate both concepts (precision and readability) and answered with the same value for each one in all of the tests. For this reason, in analyzing each word and sentence, we will not take into account each concept separately, and will instead analyze them all at once.

Let us start with *W1* – Cristina. This word was chosen to test dactilology alone (fingerspelling), since names are signed by fingerspelling. The biggest critiques were about the speed of the signing being too fast. The used speed was the default one from HamNoSys. In JaSigning, we can slow down the speed of the signing. We can also slow it down within the HamNoSys with a slow character on the movement, but this may be too slow for someone reading. As such, the best option is set JaSigning’s signing speed to a lower value. Another answer told us that the dactilology is signed at chest level and that some signs were being signed below it. At the time of annotation, the default location was used (no location was
Figure 5.9: Answers to all of the words and sentence’s precision and readability. The \( y \)-axis represents the answer value range from 1 – 5, and the \( x \)-axis the words and sentences from 1 to 5. T1, T2 and T3 stand for Testee 1, 2 and 3.

specified). \textit{JaSigning} appears to use the default location as being in front of the belly, and given that information, all of the dactilology locations were changed to being in front of the chest afterwards.

The second word \( W_2 \), Lisboa, was understood by all and well read. However, two testees answered \textit{Lisboa or Julho}, since both signs are similar. \( W_3 \) – Carro – was also easily recognized and read.

\( W_4 \), Mapa, was harder to recognize. The testee that knew LGP at a native level said that she thought that it was a folding screen, and that there was another way to sign Mapa. In \textit{SpreadTheSign}, there is only the way it was presented in the evaluation. This is important because it tells us that there is a limit up to where a normal person can annotate words into the system. The dictionary does not contain all words, and for the words it has, it does not contain all variations of those. The system has to be annotated by experts in order to be more reliable.

With \( W_5 \), ir, the pulse rotation was the toughest thing to decipher by the testees. However, 2 of the 3 testees could still recognize the word.

Hopping onto the first sentence \( S_1 \), \textit{O Tiago come a sopa.}, the bigger problem was once again the dactilology’s speed being too fast. On the second sentence \( S_2 \), \textit{Boa tarde.}, which is made up of the signs \textit{BOM} + \textit{TARDE}, the sign \textit{BOM} was also said to be a bit too fast, making \textit{JaSigning}'s default speed not only too fast for dactilology but also for regular signs.

Sentence \( S_3 \), \textit{O Ricardo não escreveu.}, presented a negative and a past tense. The past tense was perceivable, but for one of the testees it could be more natural. The non-manual expression was also said to be good.

Sentence \( S_4 \) was a failure. Due to a mistake, the sentence used was not the sentence signed. As
such this sentence is discarded from the tests. As for sentence S5, Tu treinas o cão?, it concludes again that the experts need to do the annotation. The testees did not understand the word treinar, which was annotated directly from SpreadTheSign.

We can sum up the test's results all together: the default signing speed of this avatar is too fast, which can be directly slowed down on JaSigning's options, and annotating purely with SpreadTheSign's help may not be enough for a non-expert to annotate.
Conclusions and Future Work

Contents

6.1 Project’s Conclusions ....................................................... 65
6.2 Future Work ................................................................. 66
6.1 Project’s Conclusions

The topic over which this thesis picks up on is, to our surprise, very popular among studies. Its importance is backed up by plenty of research around the world, and that plays a big role in captivating researchers into dwelling onto this topic. Moreover, there is still so much to study for each Sign Language that each country should become busy over them in the next decades: the motivation for Sign Language translation projects is very strong.

It is important to state that these systems offer, just like common Spoken Language to Spoken Language translators, a helping hand in short utterances, and not perfect translations over full unpredictable texts and books. This is important for people understanding the pros of researching these systems, and also for evaluating these systems.

This system could be practical for this simple usage: the fact that everyone has the ability to annotate words into the dictionary if they want to, without a big learning curve, really makes it a system for everyone to use on all ends. Even if this thesis only has, in total, 63 syntax rules, people can also add them. It should also be fairly easy for specialists in LGP to annotate the signs themselves, for a better precision and capturing a more authentic animation (i.e: an animation as close as it can be to the actual signing).

Another interesting thing is that this model of a system can be used for any sign language out there: the only things that differ are the linguistics and the rules attached to them. The dictionary also changes, but the annotation method is still precisely the same.

Every work done about this topic seems to capture the essence of a step in the right direction, and hopefully, with this thesis, the translation of Portuguese to LGP is now a step closer to its goal than before.
6.2 Future Work

The first thing that should be noted is that this project uses an existing avatar (*JaSigning*) for the reproduction of the signs. As such, some possible future work could dwell with the creation of a new and better avatar that can sign the output of this project, with smooth movements and taking a bigger approach on computer graphics.

Besides adding another module to the project, more should be added to the current one. Morphologically speaking, adding a compatibility to all types of subjects will be a good addition to the system. Analyzing more Spacy tags and studying how they can improve our translation is also something that can be done. Since everyone is able to annotate and add syntactic rules to the system, another interesting work that can be done is adding an interface that allows users to easily input new rules into the system, or even to update their system’s rules according to someone’s rules, creating different rules presets - this could be used for different user’s preferences or even different Sign Languages.

Something else that is important and that was not done in this project is related to the fact that it does not deal with co-references. As such, future work may focus in this matter.

Finally, it is unfortunate that we could not use AZee as it was not yet implemented - it would have allowed for the annotation of timed non-manual expressions, improving the naturalness of the sign movements’ reproduction in the avatar and introducing multi-modality into our system. When it becomes available, it would be interesting to study AZee’s approach and see how it would improve this system. If AZee still cannot be used then, the fact that SiGML is an XML-based language can still be taken advantage of, and timestamps, for instance, can be added accordingly.
Bibliography


A.1 Directory Distribution

The root of the project (located at the start of the main directory) contains only four files: first_module.py, second_module.py, main.py (which joins both modules together) and README.md, which is a basic README file that can be consulted in order to get more information on how to install necessary packages and on how to run the project. The file first_module.py is related to the Sign Tokenization Module (Section 4.1.2) and the file second_module.py is related to the SiGML Generation Module (Section 4.1.3). Besides the files that are located on the root folder, there are directories which hold the rest of the information:

- **doc**: directory related to the writing of the thesis’ document;
- **util**: contains python files that implement util functions, mostly Spacy functions;
- **EBSE**: stands for Example Based Syntax Exchange and contains syntax_exchange.py, which deals with the syntax exchange operation from Portuguese to LGP. Also contains the .csv files that have in them the syntax translation rules:
• **outputs**: whenever any of the modules outputs something into a file, those files are located in this directory;

• **tests**: contains a few scripts used for testing (not to be confused with evaluation testing);

• **dictionary**: contains the available .sigml files that are part of our system’s dictionary;

• **annotations**: contains files with the used syntax annotations.

### A.2 Main Script and its Modules

Since we are working with **Python**, the main script can be found within the root of the directory as `main.py`, and it can be ran in the terminal using `python main.py`. A `verbose` mode is also included, which runs the code with much needed prints in times of debugging and understanding many intermediate steps of the code. To run the code with `verbose`, simply add the argument `-v` in any part of the main call (ex: `python main.py -v`). There is one final thing this call needs, and that is an input. We are doing an European Portuguese text to an LGP 3D avatar translation, and as such, we need to pass the text that needs to be translated as input in the terminal as well. More than a sentence can be passed in a call. Since the input is in text format, it represents the variable type `STRING`, which means that it needs to be put in between quotation marks (ex: `python main.py “O João come a sopa. A Maria come carne.” -v`). With a call in this pattern, `main.py` is ready to start executing its code.

### A.3 Sentence Tokenization

**NLTK** allows us to tokenize sentences in two ways: training our sentence tokenization model, or loading an existing **NLTK** model. Loading an existing model is much faster than generating one “on the go”, which is why this option was chosen. Code Listing A.1 shows how to retrieve this model.

```python
1 stok = nltk.data.load('tokenizers/punkt/portuguese.pickle')
2 return stok.tokenize(sentences)
```

**Listing A.1**: Code used in order to load NLTK’s already existing sentence segmentation model.

In the **tests** directory, `sentence_segmentation_time_test.py` can be run in order to see this time difference. Both methods are applied and the time that each takes is printed on the console (in order to run it, type `python -m tests.sentence_segmentation_time_tests` on your terminal).
A.4 Spacy’s Basics

The first line of code that needs to be assigned for Spacy to work is the language model. Spacy requires a model in order to do the rest, and one of the models that it provides is a Portuguese model (as seen in the first line of Listing A.2).

```
1 nlp = spacy.load('pt_core_news_sm')
2 doc = nlp("O Joao come a sopa.")
```

Listing A.2: Line of code that defines the model that Spacy will be using.

This model is saved on a nlp variable (related to natural language processing), and from it, we can tokenize any sentence that we want. This is done in the second line of Listing A.2, storing the tokenization in the variable doc, which is now a list of the sentence’s tokens. When a spacy doc variable is created, all the things that we need to know from the sentence and that spacy provides are created, and can be accessed through it. These things include parts of speech, syntax dependency tags, lemmas, word shapes, stopwords or named entities.

A.5 Parts-of-Speech and Tag Map

The POS tags can be retrieved through token.pos, within the code. Exemplifying, if we want the part of speech of the first token, we can simply type `doc[0].dep` (remember `doc` from Section 4.1.2.B).

In Section 4.1.2.C, we say that we use regular expressions to gather matches between `<` and `>`. The regular expression that we use is shown in Listing A.3.

```
1 pattern = re.compile(r'\|?(<[A-Za-z\-_]+>|\|)\|?')
2 matches = pattern.findall(tag)
3 return matches
```

Listing A.3: Code that returns the tags between `<` and `>`. "

Analyzing List. A.3, we can tell that the regular expression finds the matches between `<` and `>` with a word inside of it ([A-Za-z]+). These tags, which the regular expression gathers, represent additional information on the PoS tags provided by Spacy. As for the other tags, we get all of the tokens that do not start or end in `<` and `>`, as seen in List. A.4.

\footnotetext{1}All of these things can be found here: https://spacy.io/usage/linguistic-features
Since there is a fair amount of annotations and it would be impractical to look through all of these in order to know what tags there are, a script called `get_all_tags.py` was created in order to understand how many and what morphological main classes we can get from the tag map, in the `tests` directory (it can be run typing `python -m tests.get_all_tags` in a terminal). The resulting tags can be found in Table C.1.

The same thing was done for the tags that are between `<` and `>`. In the script `get_all_tags.py` a list with all possible of these tags is shown.

### A.6 Syntactic Dependencies

These can be obtained through `token.dep_`, in the code. For instance, if we want the dependency of the first token, we can just use `doc[0].dep_` (once again, remember `doc` from Section A.4).
### List of Annotated Sentences

<table>
<thead>
<tr>
<th>EU-PT Sentences</th>
<th>LGP Syntax Change Annotations (by LGP specialist):</th>
</tr>
</thead>
<tbody>
<tr>
<td>O João come a sopa.</td>
<td>J O Â O + SOPA + COMER ou J O Â O + SOPA + (Classificador COLHER/COMER)</td>
</tr>
<tr>
<td>O João come a sopa?</td>
<td>J O Â O + SOPA + COMER (Expressão Interrogativa) ou J O Â O + SOPA + (Classificador COLHER/COMER) (Expressão Interrogativa)</td>
</tr>
<tr>
<td>O João comeu a sopa.</td>
<td>J O Â O + SOPA + COMER (Expressão ENM - Passado) ou J O Â O + SOPA + (Classificador COLHER/COMER) (Expressão ENM - Passado)</td>
</tr>
<tr>
<td>O João comerá a sopa.</td>
<td>J O Â O + SOPA + COMER + “IR VA VA” (Expressão ENM) ou J O Â O + SOPA + (Classificador COLHER/COMER) + “IR VA VA” (Expressão ENM)</td>
</tr>
<tr>
<td>O João vai comer a sopa?</td>
<td>J O Â O + SOPA + COMER (Expressão Interrogativa) ou J O Â O + SOPA + (Classificador COLHER/COMER) (Expressão Interrogativa)</td>
</tr>
<tr>
<td>Vais comer a sopa?</td>
<td>SOPA + COMER + TU (Expressão Interrogativa) ou SOPA + COMER (Classificador COLHER/-COMER) + TU (Expressão Interrogativa)</td>
</tr>
<tr>
<td>O João come.</td>
<td>J O Â O + COMER (Expressão e movimento de cabeça afirmativa)</td>
</tr>
<tr>
<td>EU-PT Sentences:</td>
<td>LGP Syntax Change Annotations (by LGP specialist):</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>O João não come a sopa.</td>
<td>J O À O + SOPA + COMER + NÃO (Expressão negativa + indicador + movimento leve de cabeça negativa) ou J O À O + SOPA + COMER (Classificador COLHER/COMER) + NÃO (= anterior)</td>
</tr>
<tr>
<td>O João não come sopa.</td>
<td></td>
</tr>
<tr>
<td>O João come a sopa com a colher.</td>
<td>J O À O + SOPA + COMER + O QUE (Expressão interrogativa+ movimento leve de cabeça) + COLHER(Classificador COLHER/COMER)</td>
</tr>
<tr>
<td>O João come a sopa da Sofia.</td>
<td>J O À O + SOPA + (MULHER + INDEX + S O F I A) + (descrição da tigela de sopa + localização receptor para emissor) + (classificador COLHER/COMER)</td>
</tr>
<tr>
<td>O João come a sopa da Sofia com a colher.</td>
<td>J O À O + SOPA + (MULHER + INDEX + S O F I A) + (descrição da tigela de sopa + localização receptor para emissor) + (classificador COLHER/COMER 2X “mostrar a colher ao alto”) + CL COLHER/COMER</td>
</tr>
<tr>
<td>O João come a sopa da Sofia com a colher vermelha.</td>
<td>J O À O + SOPA + (MULHER + INDEX + S O F I A) + (descrição da tigela de sopa + localização receptor para emissor) + (classificador COLHER/COMER) + INDEX M2 para COLHER + VERMELHO</td>
</tr>
<tr>
<td>O João, frustrado, não comeu a sopa.</td>
<td>(J O À O + INDEX) + FRUSTRAÇÃO pausa + SOPA + COMER + NÃO (Expressão negativa + “0” + movimento leve de cabeça negativa + ENM “shsh”)</td>
</tr>
<tr>
<td>A sopa é comida pelo João.</td>
<td>(SOPA + INDEX) + (COMER + QUEM) + gesto comer ao mesmo tempo abanar a cabeça p pergunta + J O À O + INDEX + COMER (CL COLHER)</td>
</tr>
<tr>
<td>A sopa foi comida pelo João.</td>
<td>(SOPA + INDEX+ ACABAR (conf. inicial 54+ conf. final 58) + QUEM + J O À O + INDEX + COMER (CL COLHER)</td>
</tr>
<tr>
<td>A sopa vai ser comida pelo João.</td>
<td>(SOPA + INDEX) + (J O À O + INDEX) + COMER + (BREVE 2X ENM va va) SOPA + (SABOR + conf 58 + expr facial) + SAL + MAIS + CL saleiro)</td>
</tr>
<tr>
<td>A sopa está insonsa, precisa de sal.</td>
<td>SOPA + (SABOR + conf 58 + expr facial) + SAL + MAIS + CL saleiro) - pausa no “e” com as duas mãos</td>
</tr>
<tr>
<td>O João, aluno da Ana, comeu a sopa.</td>
<td>(J O À O + INDEX) + (ALUNO + A N A + PRON POSS) + SOPA + COMER (CL colher ou só verbo comer + ENM “sh sh”)</td>
</tr>
<tr>
<td>João, come a sopa!</td>
<td>J O À O + SOPA + COMER + INDEX palma da mão + expr facial mais carregada)</td>
</tr>
<tr>
<td>O João tem almoçado sopa.</td>
<td>ALMOÇO + J O À O + SOPA + COMER + linha do tempo (passado ate ao momento+ ENM “sopro”)</td>
</tr>
<tr>
<td>EU-PT Sentences:</td>
<td>LGP Syntax Change Annotations (by LGP specialist):</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>O João sujou-se enquanto comia sopa.</td>
<td>J O À O + (por enquanto Conf 54 na orelha 2x) + SOPA + (COMER CL COLHER) + SUJAR peito ENM</td>
</tr>
<tr>
<td>Recentemente, o João provou sopa de legumes.</td>
<td>(RECENTEMENTE há pouco tempo ENM) + J O À O + SOPA + (LEGUMES + VÁRIOS) + COMER 2x ENM</td>
</tr>
<tr>
<td>O João comeu a sopa rapidamente.</td>
<td>J O À O + SOPA + (COMER CL COLHER movimento rápido+ENM)</td>
</tr>
<tr>
<td>O João achou a sopa deliciosa.</td>
<td>J O À O + (achar SABOR mov ligeiro de cabeça 2 a 3x até ao fim da frase) + SOPA + (BOA 2x ENM)</td>
</tr>
<tr>
<td>João, preciso que compres uma sopa.</td>
<td>(J O À O + CHAMAR 2x) + PRECISAR + (TU INDEX) + SOPA + (COMPRAR ENM)</td>
</tr>
<tr>
<td>A sopa foi comprada pelo João.</td>
<td>(SOPA + INDEX) + (COMPRAR + QUEM) + (J O À O + INDEX)</td>
</tr>
<tr>
<td>O João comeu a maçã.</td>
<td>J O À O + MAÇA + COMER CL formato da maçã</td>
</tr>
<tr>
<td>O João bebe a sopa.</td>
<td>J O À O + SOPA + (BEBER CL formato tigela)</td>
</tr>
<tr>
<td>Ontem aprendi inglês.</td>
<td>ONTEM + EU + INGLÊS + APRENDER ENM &quot;sh sh&quot;)</td>
</tr>
<tr>
<td>Ontem, eu aprendi inglês.</td>
<td>(ONTEM + pausa) + EU + INGLÊS + APRENDER ENM &quot;sh sh&quot;)</td>
</tr>
<tr>
<td>Eu aprendo inglês no quarto.</td>
<td>QUARTO + EU + INGLÊS + APRENDER ENM &quot;m m&quot; ou EU + INGLÊS + APRENDER ENM &quot;m m&quot; + QUARTO</td>
</tr>
<tr>
<td>A festa vai ser no pátio da faculdade.</td>
<td>FESTA + UNIVERSIDADE + FORA + ESPAÇO CERVEJA + BARATA + SABOR IX(BOCA) + NÃO PRESTAR ou MÁ (configuração &quot;B&quot;)</td>
</tr>
<tr>
<td>Apesar de barata, a cerveja é má.</td>
<td>BARATA + VERDADE ENM &quot;vvv&quot; + CERVEJA + NÃO PRESTAR ou MÁ (configuração &quot;B&quot;)</td>
</tr>
<tr>
<td>O jantar estava muito agradável.</td>
<td>NOITE + COMER + COMER + BOM 2x</td>
</tr>
<tr>
<td>O jantar estava deliciosíssimo.</td>
<td>NOITE + JANTAR + DOCE+BOM 2x (configuração bico de pato)</td>
</tr>
<tr>
<td>Os ingredientes são: sal, carne, pimenta e alho.</td>
<td>COMIDA + O QUE + SAL + CARNE + PI- MENTA + ALHO (mão esquerda aberta e apontar para cada dedo, a começar pelo indicador=sal, médio=carne, anelar=pimenta, mínimo=alho)</td>
</tr>
<tr>
<td>Não faças isso!</td>
<td>FAZER + NÃO (dentes cerrados) ou NÃO (dentes cerrados)</td>
</tr>
<tr>
<td>O Benfica perdeu!</td>
<td>BENFICA + PERDER</td>
</tr>
<tr>
<td>- Faz isso. - disse o Tiago.</td>
<td>FAZER + palma da mão virada p cima e em frente + APONTAR (para ele) + T I A G O + DIZER (configuração indicador)</td>
</tr>
<tr>
<td>O Romão foi de Lisboa ao Porto.</td>
<td>R O M Á O + LISBOA + PORTO + IR (localização de Lisboa para Porto)</td>
</tr>
<tr>
<td>Ele é de uma grande simpatia.</td>
<td>ELE + SIMPÁTICO (expressão facial)</td>
</tr>
<tr>
<td>O bando de aves migra na Primavera.</td>
<td>AVES + GRUPO + PRIMAVERA + VOAR + IR</td>
</tr>
<tr>
<td>A multidão afastou-se do local.</td>
<td>PESSOAS + MUITAS (Classificador) + AFAS- TAR (classificador)</td>
</tr>
<tr>
<td>EU-PT Sentences:</td>
<td>LGP Syntax Change Annotations (by LGP specialist):</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>O João é simpático</td>
<td>J O A O + SIMPÁTICO</td>
</tr>
<tr>
<td>O João e a Ana são simpáticos.</td>
<td>J O A O + ANA + DOIS (apontar para eles) + SIMPÁTICO</td>
</tr>
<tr>
<td>O João foi vítima de um roubo.</td>
<td>J O AO + PREJUDICADO + ROUBAR + ELE</td>
</tr>
<tr>
<td>O João é mais forte que o Tiago.</td>
<td>J O AO + APONTAR + FORTE + T I A G O + DO QUE ELE</td>
</tr>
<tr>
<td>O João é maior que o Tiago.</td>
<td>J O AO + APONTAR + MAIS ALTO + T I A G O + DO QUE ELE</td>
</tr>
<tr>
<td>O João é tão humilde como o Tiago.</td>
<td>J O AO + APONTAR + SIMPLES + T I A G O + DOIS + IGUAL (comparação)</td>
</tr>
<tr>
<td>O João é menos alto que o Tiago.</td>
<td>J O AO + APONTAR + PEQUENO + T I A G O + APONTAR + ALTO + DO QUE</td>
</tr>
<tr>
<td>O João é menor que o Tiago.</td>
<td>J O AO + APONTAR + PEQUENO + T I A G O + APONTAR + ALTO + DO QUE</td>
</tr>
<tr>
<td>A Ana, professora de faculdade, deu uma aula hoje.</td>
<td>A N A + APONTAR (pausa) + PROFESSOR + UNIVERSIDADE + AULA + HOJE + ENSINAR</td>
</tr>
<tr>
<td>O João comprou uma cenoura e três maçãs.</td>
<td>J O A O + COMPRAR + CENOURA + UM + MACA + TRÉS</td>
</tr>
<tr>
<td>O João comprou duas vezes mais maçãs do que cenouras.</td>
<td>J O A O + MACA + COMPRAR + DOBRO + CENOURA + MENOS</td>
</tr>
<tr>
<td>O João comprou meia dúzia de batatas.</td>
<td>J O A O + COMPRAR + BATATAS + QUANTIDADE + SEIS</td>
</tr>
<tr>
<td>A Índia foi descoberta por mar.</td>
<td>INDIA + DESCOBRIR + ABANAR A CABEÇA (pergunta) + MAR + TRAJETO</td>
</tr>
<tr>
<td>Faz o trabalho de casa, Ana!</td>
<td>TRABALHO + CASA + FAZER + palma da mão virada p cima e em frente</td>
</tr>
<tr>
<td>Seria ideal que eu tivesse boa nota.</td>
<td>IDEAL + SE + EU + NOTA + BOA</td>
</tr>
<tr>
<td>Ah! Já entendi!</td>
<td>AH + PERCEBER</td>
</tr>
<tr>
<td>Ai! Isso magou-me.</td>
<td>AI + ISSO (apontar) + DOER + EU (a mim)</td>
</tr>
<tr>
<td>A coelhinha deixou o ovo da Páscoa no meio do arbusto.</td>
<td>COELHO + MULHER + PEQUENA + PASCOA + OVO + ARVORE PEQUENA + PÔR</td>
</tr>
<tr>
<td>O coelhinho colheu o ovo da coelhinha, e comeu-o.</td>
<td>COELHO + PEQUENHO + OVO + MULHER + COELHO + PEQUENA + APANHAR + COMER</td>
</tr>
<tr>
<td>O rapaz da aldeia tentou encontrar o ovo da coelhinha, sem sucesso.</td>
<td>RAPAZ + ALDEIA + ÁREA + OVO + TENTAR + PROCURAR + NADA + NÃO CONSEGUIR</td>
</tr>
<tr>
<td>Olá, bom dia.</td>
<td>OLA + BOM + DIA</td>
</tr>
<tr>
<td>Boa tarde.</td>
<td>BOA + TARDE</td>
</tr>
<tr>
<td>Como estás?</td>
<td>SAUDE + BEM?</td>
</tr>
<tr>
<td>Olá, como tens passado?</td>
<td>OLÁ + SAÚDE + BEM + LONGO DO TEMPO?</td>
</tr>
<tr>
<td>Ora viva!</td>
<td>OLÁ!</td>
</tr>
<tr>
<td>Está tudo bem contigo, João?</td>
<td>SAUDE + BEM + TU (apontar) não é preciso mencionar o nome “João”</td>
</tr>
<tr>
<td>João, passa-se alguma coisa?</td>
<td>ACENAR (ao João) + COISA + PASSAR-SE 2x (com 2 mãos)?</td>
</tr>
<tr>
<td>Boa noite, amigos.</td>
<td>BOA + NOITE + AMIGOS + ÁREA</td>
</tr>
<tr>
<td>Saudações, irmãos!</td>
<td>não costumamos dizer isso</td>
</tr>
<tr>
<td>Muito prazer em conhecê-lo.</td>
<td>CONHECER + PESSOA + PRAZER (peito)</td>
</tr>
<tr>
<td>É um enorme prazer conhecê-la.</td>
<td>CONHECER + PESSOA + PRAZER (peito) + BOM (configuração bico de pato)</td>
</tr>
<tr>
<td>EU-PT Sentences:</td>
<td>LGP Syntax Change Annotations (by LGP specialist):</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Por favor, sinta-se em casa!</td>
<td>POR FAVOR + SENTIR + DENTRO (interior) + IGUAL + CASA</td>
</tr>
<tr>
<td>O prazer é todo meu.</td>
<td>PRAZER (peito) + MEU (inclinar a cabeça para à frente = respeito)</td>
</tr>
<tr>
<td>Sendo assim, adeus!</td>
<td>ESPANTO (com duas mãos) + ADEUS</td>
</tr>
<tr>
<td>Até logo, João!</td>
<td>ATE LOGO + ADEUS não é preciso mencionar o nome “João”)</td>
</tr>
<tr>
<td>Vejo-vos mais tarde no café.</td>
<td>ENCONTRAR + CAFE + MAIS TARDE</td>
</tr>
<tr>
<td>O cavalo correu pelo prado.</td>
<td>CAVALO + CAMPO (descrição) + CORRER (descrição de 4 patas - classificador)</td>
</tr>
<tr>
<td>O pônei correu pelo prado.</td>
<td>CAVALO + PEQUENO + CAMPO (descrição) + CORRER (descrição de 4 patas - classificador)</td>
</tr>
<tr>
<td>A professora que ensina inglês faltou hoje.</td>
<td>PROFESSORA + TAL + INGLÊS + ENSINAR + FALTAR</td>
</tr>
<tr>
<td>O sino da igreja que toca todos os dias não está a funcionar.</td>
<td>IGREJA + SINO + SEMPRE + TODOS OS DIAS + FUNCIONAR-NÃO</td>
</tr>
<tr>
<td>O patrão mencionou que o prazo é no domingo, e que temos que nos despachar.</td>
<td>PATRÃO + DIZER (só um dedo) + PRAZO + DOMINGO + ACABAR + TER (OBRIGAÇÃO) + NÓS + DESPACHAR</td>
</tr>
<tr>
<td>O pranto, que era tão verde como o Verão, dava uma sensação de encanto.</td>
<td>CL PAISAGEM + VERDE + REALCE + TAL COMO + VERÃO + DAR-ME + SENTIR + BEM</td>
</tr>
<tr>
<td>Auxílio quem te ajuda.</td>
<td>TU + AJUDAR + SÓ SE + QUEM + AJUDAR-TE</td>
</tr>
<tr>
<td>Subitamente, o rapaz apercebeu-se aquilo não fazia sentido.</td>
<td>DE REPENTE + RAPAZ + PERCEBER (localização no nariz) = ESPANTO + ISSO + SENTIDO + NÃO-HAVER</td>
</tr>
<tr>
<td>Através de uma lupa, o pequeno sapo analisou o nenúfar.</td>
<td>LUPA + SAPO + PEQUENO + LUPA + FLOR + ÁGUA + FLOR CL + LUPA (continuidade)</td>
</tr>
<tr>
<td>Todos os dias, a Ana lava os dentes.</td>
<td>AMANHA + AMANHA + A N A + LAVAR DENTES</td>
</tr>
<tr>
<td>De cinco em cinco dias, o João arranja o cabelo.</td>
<td>DIAS + 5 DIAS + 5 + J O À O + CABELO + ARANJAR</td>
</tr>
<tr>
<td>Portugal nunca viveu uma situação como estas.</td>
<td>PORTUGAL + ANTES + SITUAÇÃO + IGUAL + ISSO + NUNCA</td>
</tr>
<tr>
<td>Não penso dessa forma porque me parece prejudicial.</td>
<td>EU + PENSAR (opinião) + ISSO + NÃO + ACHAR + PREJUÍZO + DAR</td>
</tr>
<tr>
<td>Visto que me parece prejudicial, não penso dessa forma.</td>
<td>EU + ACHAR + PREJUÍZO + ISSO + EU + OPINIÃO + IGUAL + NÃO</td>
</tr>
<tr>
<td>O rapaz, cujo nome é longo, não se apresentou.</td>
<td>RAPAZ + NOME + DELE + GRANDE + QUEM + PESSOAL + APRESENTAR-NÃO</td>
</tr>
<tr>
<td>Naquele momento, ela só queria que o tempo passasse.</td>
<td>TEMPO + APONTAR + ELA + QUERER (vá vá) + TEMPO + PASSAR</td>
</tr>
<tr>
<td>Muito obrigado pela ajuda!</td>
<td>MUITO + OBRIGADA + AJUDA</td>
</tr>
</tbody>
</table>
C.1 Spacy’s Tag Map - Possible Tag Patterns

- ADJ [gender, number]
- ADV [gender, number]
- ART [gender, number]
- DET [gender, number]
- IN [ ]
- INDP [gender, number]
- KC [ ]
- KS [ ]
- N [gender, number]
C.2 Spacy’s Tag Map - Tags between < and >

- `<NUM-fract>`: A fractional number;
- `<NUM-ord>`: Ordinal numbers (sétimo, oitavo, etc);
- `<SUP>`: Superlative degree of an adjective;
- `<atemp>`: Temporal adverb;
- `<card>`: A cardinal number;
- `<dem>`: A demonstrative pronoun (este, esta, etc);
- `<foreign>`: A foreign word (in portuguese, estrangeirismo);
- `<interr>`: Interrogative determinants (quanto, quem, porquê).
### C.3 Spacy’s Tag Map Morphological Classes

<table>
<thead>
<tr>
<th>Morph. Class</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>Adjective [Adjetivo]</td>
</tr>
<tr>
<td>ADP</td>
<td>Adposition [?]</td>
</tr>
<tr>
<td>ADV</td>
<td>Adverb [Adverbio]</td>
</tr>
<tr>
<td>ART</td>
<td>Article [Artigo]</td>
</tr>
<tr>
<td>CONJ</td>
<td>Conjunction [Pontuação]</td>
</tr>
<tr>
<td>DET</td>
<td>Determinant [Determinante]</td>
</tr>
<tr>
<td>EC</td>
<td>? [?]</td>
</tr>
<tr>
<td>IN</td>
<td>Interjection [Interjeição]</td>
</tr>
<tr>
<td>INDP</td>
<td>Independent Pronoun [Pronome Independente]</td>
</tr>
<tr>
<td>KC</td>
<td>Coordinating Conjunction [Conjunção Coordenativa]</td>
</tr>
<tr>
<td>KS</td>
<td>Subordinating Conjunction [Conjunção Subordinativa]</td>
</tr>
<tr>
<td>N</td>
<td>Noun [Nome Comum]</td>
</tr>
<tr>
<td>NOUN</td>
<td>? [?]</td>
</tr>
<tr>
<td>NUM</td>
<td>Number [Número]</td>
</tr>
<tr>
<td>PERS</td>
<td>Personal Pronoun [Pronome Pessoal]</td>
</tr>
<tr>
<td>PROP</td>
<td>? [?]</td>
</tr>
<tr>
<td>PROPN</td>
<td>Proper Noun [Nome Propio]</td>
</tr>
<tr>
<td>PRP</td>
<td>Preposition [Preposição]</td>
</tr>
<tr>
<td>PU</td>
<td>Punctuation [Pontuação]</td>
</tr>
<tr>
<td>V</td>
<td>Verb [Verbo]</td>
</tr>
<tr>
<td>X</td>
<td>Other [Outros]</td>
</tr>
<tr>
<td>_</td>
<td>Underscore [Underscore]</td>
</tr>
<tr>
<td>_SP</td>
<td>Space [Space]</td>
</tr>
</tbody>
</table>

**Table C.1:** Table depicting Spacy’s tag map morphological classes. There are a few more, but they are duplicates without being in capital lettering (for instance, besides ADJ there is adj). The question marks (marked as ? [?]) represent unknown meanings: these were classified as such only after not finding any guides or e-mailing Spacy’s team on these meanings and not getting an answer back.
Evaluation Appendix

D.1 Syntax Evaluation Sentences
### Random Sentences:

| A Beatriz foi à praia.                | A Beatriz lava o chão.                |
| O Nuno trabalhou muito.             | O Ricardo trabalha o campo?          |
| A Rute fechou os estores.           | A Sofia melhorou a situação.         |
| O Gonçalo comprou comida.           | O Simão beberá a água.               |
| O Ricardo deu uma flor à Maria.     | O Rúben vai conduzir o carro?        |
| A Maria gostou muito da flor.       | Vais vender a casa?                  |
| A praia estava muito boa.           | O Pizzi treina.                      |
| As toalhas não secaram bem.         | A Ritinha não faz os deveres.        |
| A caneta está a ficar sem tinta.    | Ritinha, faz os deveres!             |
| As gomas são mais para a saúde.     | O Pizzi treina.                      |
| Felizmente tudo correu bem.         | O Ricardo afastou-se da zona.        |
| Eu vou trabalhar muito hoje.        | O Rúben e a Maria são bestiais.      |
| Fez muito calor no Alentejo.        | Quero ir almoçar contigo.            |
| O frigorífico tem esparguete e carne.| De três em três meses, o Rui faz o jantar. |
| Os iogurtes de morango são deliciosos.| Seria ideal que ele cozinhasse bom arroz. |
| Os carros apitaram efusivamente.   | O Ricardo afastou-se da zona.        |
| O Mantorras marcou golo.            | A Rita é menos estudiosa que a Sofia.|
| Estás um calor infernal.            | O Grande foi de Aveiro ao Alentejo.  |
| O almoço está bom?                  | A Carla é mais estudiosa que a Rita.|
| Vamos almoçar onde?                 | A Sofia é mais estudiosa que a Rita.|
| Tirei a licenciatura noutra faculdade.| A Rita é tão estudiosa a Sofia.      |
| O primeiro lugar é do Phelps.       | O Montepio falou!                    |
| O telemóvel precisa de carregar.    | O Diogo, director da empresa, ditou uma palestra ontem.|
| Temos de comprar mais sabonete.     | A professora explicou a matéria sucintamente.|
| O João faz anos no dia 14.          | A Rita é menor que a Sofia.          |
| Queres que faça um bolo?            | O Diogo faz o trabalho da Sofia.     |
| Algém roubou a pastelaria do António.| O Diogo faz o trabalho da Sofia com o lápis. |
| O Alentejo é um lugar mágico!       | As ferramentas são: martelo, tesoura, régua e lápis. |
| Como estás, João?                   | A professora explicou a matéria sucintamente.|
| A professora deu matéria nova.      | A Diogo faz exercício físico.        |
| A matéria nova é interessante.     | O Diogo fez exercício.               |
| Este tratamento é demasiado caro.   | O Diogo trabalhou!                   |
| Tens dinheiro que me emprestes?     | O Diogo se envergonhou.              |
| Hoje vamos comer num restaurante.   | A Rita é menor que a Sofia.          |
| A banda tocou maravilhosamente.     | Quero ir almoçar contigo.            |
| O ritmo desta dança é estupendo.    | A Rita é tão estudiosa a Sofia.      |
| Ela escreveu-me uma carta.          | As ferramentas são: martelo, tesoura, régua e lápis. |
| Tira as mãos da comida, José!       | A Rita é mais estudiosa que a Sofia.|
| O pêssego é sumarento.              | O Diogo fez exercício.               |
| As frases estão a ser avaliadas.    | A Rita é menor que a Sofia.          |

### 'In Accordance to Syntax Rules' Sentences:

| A Beatriz foi à praia.                | A Beatriz lava o chão.                |
| O Nuno trabalhou muito.             | O Ricardo trabalha o campo?          |
| A Rute fechou os estores.           | A Sofia melhorou a situação.         |
| O Gonçalo comprou comida.           | O Simão beberá a água.               |
| O Ricardo deu uma flor à Maria.     | O Rúben vai conduzir o carro?        |
| A Maria gostou muito da flor.       | Vais vender a casa?                  |
| A praia estava muito boa.           | O Pizzi treina.                      |
| As toalhas não secaram bem.         | A Ritinha não faz os deveres.        |
| A caneta está a ficar sem tinta.    | Ritinha, faz os deveres!             |
| As gomas são mais para a saúde.     | O Pizzi treina.                      |
| Felizmente tudo correu bem.         | O Ricardo afastou-se da zona.        |
| Eu vou trabalhar muito hoje.        | O evento vai ocorrer no quarto da casa. |
| Fez muito calor no Alentejo.        | O Montepio falou!                    |
| O frigorífico tem esparguete e carne.| De três em três meses, o Rui faz o jantar. |
| Os iogurtes de morango são deliciosos.| Seria ideal que ele cozinhasse bom arroz. |
| Os carros apitaram efusivamente.   | O Ricardo afastou-se da zona.        |
| O Mantorras marcou golo.            | A Rita é menos estudiosa que a Sofia.|
| A qualidade desta câmara é excelente!| A Rita é menor que a Sofia.          |
| Está um calor infernal.             | O Diogo, director da empresa, ditou uma palestra ontem.|
| A impressora é velha, mas funcional.| O rapaz bebeu pela palhinha.         |
| A chave do carro está sem pilha.    | O António bebeu pela palhinha.       |
| O telemóvel precisa de carregar.    | O Diogo fez o trabalho da Sofia.     |
| Temos de comprar mais sabonete.     | A professora explicou a matéria sucintamente.|
| Queres que faça um bolo?            | O Diogo faz o trabalho da Sofia.     |
| Alguém roubou a pastelaria do António.| O Diogo faz o trabalho da Sofia com o lápis. |
| O Alentejo é um lugar mágico!       | As ferramentas são: martelo, tesoura, régua e lápis. |
| Viajar é o que mais gosto de fazer. | A professora explicou a matéria sucintamente.|
| Como estás, João?                   | A professora explicou a matéria sucintamente.|
| A professora deu matéria nova.      | A Diogo faz exercício físico.        |
| A matéria nova é interessante.     | O Diogo fez exercício.               |
| Este tratamento é demasiado caro.   | O Diogo trabalhou!                   |
| Tens dinheiro que me emprestes?     | O Diogo se envergonhou.              |
| Hoje vamos comer num restaurante.   | A Rita é menor que a Sofia.          |
| A banda tocou maravilhosamente.     | Quero ir almoçar contigo.            |
| O ritmo desta dança é estupendo.    | A Rita é tão estudiosa a Sofia.      |
| Ela escreveu-me uma carta.          | As ferramentas são: martelo, tesoura, régua e lápis. |
| Tira as mãos da comida, José!       | A professora explicou a matéria sucintamente.|
| O pêssego é sumarento.              | O Diogo fez exercício.               |
| As frases estão a ser avaliadas.    | A Rita é menor que a Sofia.          |
PE2LGP 3.0’s Non-Manual Expressions

The following Non-Manual Expressions have been added to PE2LGP 3.0 as a consequence of analyzing the annotated syntax sentences and understanding which would be necessary. These were obtained through video annotations of experts, as non-manual elements were not available in Spreadthesign separated from the words themselves. The list is as follows:

{?} Interrogation;

{SHSH} Past;

{VAVA} Future;

{.} Affirmative head movement and expression;

{NEG} Negative light facial expression;

{FEMALE} Female gloss.