CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Syntax-based Concept Alignment for Machine Tranlation

Provisory subtitle

Master's thesis in Computer Science

Arianna Masciolini

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Syntax-based Concept Alignment for Machine Tranlation

Provisory subtitle

Arianna Masciolini

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Syntax-based Concept Alignment for Machine Tranlation
Provisory subtitle
Arianna Masciolini

Supervisor: Aarne Ranta, Department of Computer Science and Engineering
Examiner: Carl Johan Seger, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: TODO: description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2020

Syntax-based Concept Alignment for Machine Tranlation
Provisory subtitle
Arianna Masciolini
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

TODO:

Keywords: TODO: Computer, science, computer science, engineering, project, thesis.

Acknowledgements

TODO: Here, you can say thank you to your supervisor(s), company advisors and other people that supported you during (and before) your project

Arianna Masciolini, Gothenburg, December 2020

Contents

List of Figures

List of Tables

1
Introduction

Concept Alignment (CA) consists in finding semantical correspondences between parts of multilingual parallel texts, of which the Rosetta Stone is a notable example. Such task, often preliminary to further linguistic analysis, is routinely performed by learners of classical languages when working with a translation alongside the original text. It can - and usually does happen simultaneously - at different degrees of abstraction, ranging from word to sentence level. While in some cases the student identifies new concepts thanks to language comparison itself (*concept extraction*), there are instances where a set of concepts is already known and the objective becomes finding the corresponding expressions in a certain language (*concept propagation*).

Another task that involves CA is natural language translation: the human translator, almost subconsciously, first identifies concepts in the source text and only then looks for ways to render them in the target language. It is then natural to wonder wether it is possible to make use of CA in Machine Translation (MT) as well. The hypothesis motivating this project, whose objective is to develop and test strategies for automating CA, is that it can serve as a step of a compositional MT pipeline. In such case, its users could easily be provided with a way to verify the correctness of the results by examining - at any level - the ways a concept is expressed in different languages instead of having to compare the entire text in the source language to its automatically translated counterparts. From this perspective, CA can help develop a more easily interpretable, and therefore more reliable MT system.

While MT, and in particular Statistical Machine Translation (SMT), does make use of some alignment models, traditional solutions focus on aligning words or, at most, sequences of words, thus generally putting raw strings of text in different languages in relation with each other. This makes it hard to operate at multiple levels of abstraction simultaneously. Nevertheless, there are several reasons why a system able to do that is desirable. First of all, choosing the abstraction level to operate at is not trivial, to the point that one could argue that, even within the particular context of translation, correspondences at different levels of abstraction may be more or less useful according to the specific pair of sentences they occur in. As an example, let us take the following two English-Italian pairs of sentences:

1. "*May I have a piece of cake?*" and "*Potrei avere un pezzo di torta?*"
2. "*Finding useful correspondences isn't exactly a piece of cake*" and "*Trovare corrispondenze utili non è proprio scontato*"

In the first case, correspondences on the word level, like *piece-pezzo* and *cake-torta* are definitely relevant, but in the second sentence pair *piece of cake* is used idiomat-

ically and it would probably be more useful to just put the whole phrase in relation with *scontato*.

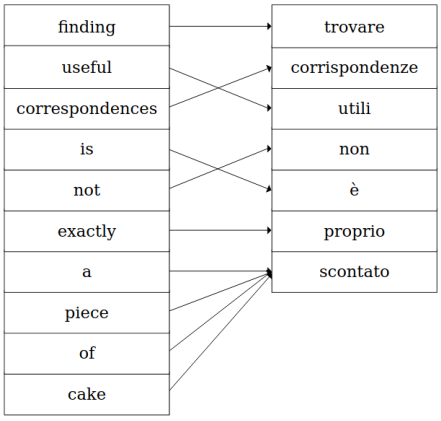


Figure 1.1: An English sentence aligned with its Italian translation. In this case, we are looking for the smallest possible alignments, but it is also possible to find higher-level correspondences, such as "*useful correspondences*" and "*corrispondenze utili*".

In cases like this, a syntactic comparison between the two sentences can help, if not to *only* extract meaningful correspondences - which would be the case in the example above, where the complement of the copula *is* (resp. *è*) is a multiword expression in English and a one-word in Italian, at least to obtain a series of alignments *at all levels* from which to select the most relevant at a later stage[1].

Furthermore, taking syntax into account allows to easily deal with non-contiguous multiword expressions, another situation which is challenging to deal with by means of purely statistical approaches. For instance, if we consider the sentence "*Without any linguistic knowledge, it is definitely hard*" and its Italian translation "*Senza conoscenze linguistiche, è decisamente arduo*", a syntactic analysis would make it possible to extract the correspondence *is hard-è arduo*, even if in both cases the copula and its complement are intercalated by an adverb.

With this thesis, we propose a syntax-based approach to both the above mentioned variants of CA - concept extraction and concept propagation - and put it to the test by integrating it in a prototype domain-specific MT system composed of a neural Universal Dependencies (UD) parser, a rule-based alignment module and a target language generation component based on Grammatical Framework (GF).

Structure of the thesis

This work is structured as follows. Chapter 1 gives a few basic definitions, including a more rigorous one of CA itself, as well as providing the necessary background and

[1] an example of a more difficult case could be the alternative translation "Trovare corrispondenze utili non è proprio *un gioco da ragazzi*", roughly corresponding to *a child's play*. In this case, trying to align the sentence based on a syntactic comparison would yield both *piece of cake-gioco da ragazzi* and the more questionable correspondences *piece-gioco* and *of cake-da ragazzi*.

contextualizing our project by reviewing a few related works. Chapter 2 and 3 focus respectively on concept extraction and propagation, presenting both our approach to each task and the corresponding experimental results, while Chapter 4 describes the prototype MT system developed to produce an overall evaluation of our CA component. Finally, Chapter 5 consists of a discussion of the overall results and our proposed ideas for future work.

2

Theory and Technologies

In this chapter, we review the preliminary notions of Computational Linguistics that are necessary for a deeper understanding of this work, give a more rigorous definition of the problem at hand and present the basic idea behind our proposed approach to automation, mentioning the technologies involved and comparing it to the existing ones.

2.1 Preliminary notions

2.1.1 Semantic compositionality

The principle of semantic compositionality states, in its most general form, that the meaning of a complex expression is determined solely by the meanings of its components and by the manner in which these components are combined [34].

A useful variant of this formulation, which refers to natural languages in particular, is the following:

Definition 1 *For every complex expression e in a language L, the meaning of e in L is determined by the structure of e and the meanings of the constituents of e in L.*

Questions of structure and constituency are settled by the syntax of L, while the meanings of simple expressions are given by the lexical semantics of L, meaning that syntax and lexical semantics, together, are sufficient to determine the entire semantics of L [34].

According to this definition, then, if L is compositional, the meaning of an expression e in L cannot depend directly on the context in which S is used in or on the intentions of the speaker who uses it, even though it might be the case that the meanings of the constituents of e depend on the context or on the speaker's intentions, thus making the sentence *indirectly* depend on them [19].

While many artificial languages, such as programming languages, are compositional by construction, the general validity of this principle for natural languages is, despite many arguments in favor, under debate. In this work, we make the assumption that compositionality also holds for natural languages, or at least that, in practice, it can be successfully applied to the vast majority of cases[1].

[1] in Section 3.2.3, however, we'll see that idiomatic expression, often used as an arguments against compositionality, are also challenging to deal with in the context of grammar-based CA.

2.1.2 Constituency grammars

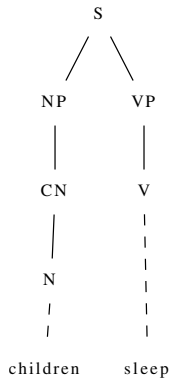
The idea of taking advantage of semantic compositionality is not new in the field of computational linguistics and traditionally involves using some kind of *constituency grammar*.

The notion of constituency or *phrase structure* grammar, introduced by Noam Chomsky in [12], is usually known to computer scientists as a class of phrase structure grammars, namely that of *Context-Free Grammars* (CFGs), that can be used to describe programming languages and plays an central role in compiler theory. As a consequence, we find giving a complete formal definition of the latter unnecessary and will restrict ourselves to an informal review of the aspects that are particularly relevant for the specific subject of this thesis.

A constituency grammar consists essentially in a set of *rewrite rules*, such as

$$S \rightarrow NP\ VP,$$

which we can read as "S can be *rewritten as* (or *substituted with*) a NP followed by a VP", or as "a sentence S is constituted by a noun phrase NP and a verb phrase VP".
Replacing the term "constituted" - as in *constituency relation* - with "composed" makes it easy to understand in which way phrase structure grammars are related to the aforementioned principle of compositionality.



$$S \rightarrow NP\ VP$$
$$NP \rightarrow CN$$
$$VP \rightarrow V$$
$$CN \rightarrow N$$
$$N \rightarrow \text{"children"}$$
$$V \rightarrow \text{"sleep"}$$

Figure 2.1: Parse tree of the simple sentence "children sleep". On the right, the minimal grammar necessary to obtain such tree.

In the context of MT, the idea of exploiting the constituency relation and, as such, compositionality, was proposed by Curry in the early 1960s [13] and first put in practice two decades later in the form of an experimental interlingual translation system, not coincidentally named Rosetta, which requires the definition of two distinct logically isomorphic *Montague grammars*[2] - one for the source language, one for the target language - and constructs an intermediate representation based on such isomorphism [23].

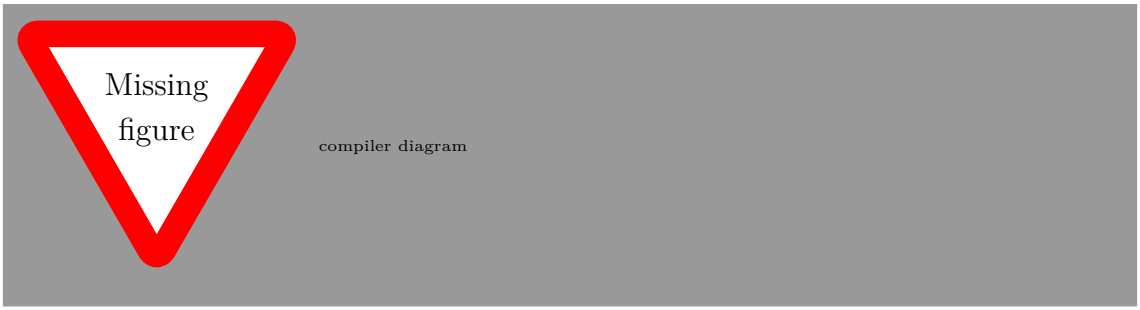
[2]Montague grammars are a semantics-oriented development of *categorial grammars*, which are in turn a type of constituency grammars.

2.1.2.1 Synchronous grammars

Among constituency grammars, other formalisms that have been employed in more recent MT systems are that of *synchronous CFG*, originally developed for programming language compilation [11] and adapted to natural language translation in several settings [36, 37, 10], and, later on, that of *synchronous TAG*, a variation of *TAG* (Tree-Adjoining Grammar) [20]. Both formalisms are meant precisely to characterize correspondences between languages by having as their elements, instead of single rewrite rules, pairs of rules - one for the source and one for the target language. In a synchronous TAG, the constituents of a source language rule may be linked to their counterparts in the corresponding target language rule, and such links, as the authors of [32] seem to suggest in section 4.1 on idiomatic expressions, may be used to identify concepts in a syntax-based fashion.

2.1.2.2 Grammatical Framework

As the repeated mention of programming language compilers in the above may suggest, it is possible to draw a very close parallel between compiler and MT pipelines: from this perspective, a natural language translation system can consist of a *frontend*, where the source language is analyzed or *parsed* and whose output is an *intermediate representation* most frequently in the form of an *Abstract Syntax Tree* (AST), and a *backend* where the AST is *linearized* by means of application of a set of rules, a process usually referred to as *code generation* in the context of programming languages and that we will refer to as target language generation or, when the context makes it clear that what the target language is, *Natural Language Generation* (NLG).



Grammatical Framework (GF), being a grammar formalism and programming language designed with this parallel in mind, goes a step further in the direction of synchronous CFGs and TAGs by introducing a clear distinction between the *abstract syntax*, whose aim is to capture the syntactic structures all natural languages taken into account have in common, and the *concrete syntaxes*, specific to each individual language, consisting in their linearization rules [26] [27]. This makes it possible to deal with multiple languages by writing only one grammar, whose components are an abstract syntax and several concrete syntaxes, providing a solid basis for a system able to translate between any pair of languages having a concrete syntax available, in any direction.

With respects to this, a GF-based MT system can be seen as an *interlingual* MT system where the intermediate representation or *interlingua* is an AST. Interlingua-

based systems have the advantage, in terms of efficiency, of making it unnecessary to build $n(n-1)$ translation functions to cover all possible pairs of n languages: having $2n$ is enough [30].

In particular, experiments with GF grammars have been conducted in the field of eXplainable Machine Translation (XMT), as the ASTs produced by source language analysis can serve both as to some extent automatically checkable certificates for the correctness of translation - by backlinearization to the source language - and as fully inspectable explanations aimed towards expert users [28].

While this approach has proved successful for domain-specific translation, i.e. in cases where the natural languages in question can be reduced to *Constrained Natural Languages* (CNLs), when it comes to open-domain translation, while target language generation remains effective, the results are negatively affected by the lack of robustness of the GF-based parsers available at the time of writing.

2.1.3 Dependency grammars

A class of grammar formalisms alternative to phrase structure, first proposed in [35], is that of *dependency grammars*, the main difference between the two being the relation they are based on. As opposed to constituency, *dependency* is a one-to-one correspondence, meaning that words are simply put in relation with each other via directed links, called in fact *dependencies*. Each dependency is, then, composed of two words: a *head* and a *dependent* that refers to it. For instance, if we try to look at syntactic dependencies in the sentence "children play", we could identify "play" as the head and "children", its subject, as its dependent.

Intuitively, this makes it so that dependency trees are simpler - in that they contain less nodes - than phrase structure trees, and their simpler structure makes them an easier target for the frontend, possibly ML-based, of a MT system such as the one outlined in the above.

Existing *dependency parsers*, such as UDPipe [33] and the Standford parser [9], are often - but not always [6] - neural pipelines trained on dependency treebanks, significantly more robust than their phrase structure counterparts. On the other hand, there is currently no effective way to use these trees as the starting point for NLG. One idea is, then, to develop a hybrid system where the frontend is a dependency parser and the backend a grammaticality-preserving GF-based target language generation module. The connecting link between these two seemingly incompatible stages of our MT pipeline can be, as we will elaborate on in Section 2.2.3.2, a CA component able to identify matching dependency trees and generate the corresponding GF concrete and abstract syntax functions: concepts.
While there are several different dependency-based frameworks, the following section focuses on the one we find most well suited to this purpose, *Universal Dependencies*.

2.1.3.1 Universal Dependencies

Universal Dependencies (UD) is a framework for cross-linguistically consistent grammatical annotation. The UD project aims at developing parallel treebanks for many languages in order to support, among other things, the development of multilingual

dependency parsers, such as the aforementioned UDPipe [33]. In order to do so, it specifies an annotation scheme and a standard format for dependency trees. The basic idea behind such standard, an evolution of (universal) Stanford dependencies [15, 14], Google universal part-of-speech tags [25], and the Interset interlingua for morphosyntactic tagsets [38], is to provide a set of POS tags, dependency relations and annotation guidelines as language-agnostic as possible - so to facilitate its application in multilingual settings - while allowing language-specific extensions whenever necessary.

In the following, we give a quick overview of the standard format UD uses to store dependency trees and of the aspects of the annotation scheme that are most relevant to the task at hand. Appendix A provides a more comprehensive, but not completely exhaustive, description of all the POS tags and dependency labels mentioned in the examples appearing in this work based on that in [29], while the reader interested in a full specification of the UD annotation scheme may refer to the official UD documentation, available at `universaldependencies.org`.

2.1.3.1.1 The CoNLL-U format

The standard plain text format for dependency trees is described in the official UD documentation [3] as an extension of CoNNL-X [8] which may contain *comment lines*, starting with `#`, *blank lines* marking sentence boundaries and *word lines* containing the annotation of a *token*[3] in 10 tab-separated fields:

1. `ID`: integer[4] representing the position of the word in the sentence, starting from 1
2. `FORM`: inflected from of the word
3. `LEMMA`: lemma of the word form
4. `UPOS`: universal POS tag
5. `XPOS`: language-specific POS tag (optional)
6. `FEATS`: list of (universal or language-specific) morphological features (optional)
7. `HEAD`: head of the current word, i.e. either the value of the `ID` of another word in the same sentence or 0 in case the word at hand is a `root`
8. `DEPREL`: universal dependency relation to the `HEAD` (`root` in case the word at hand is itself the `root`)
9. `DEPS`: enhanced dependency graph in the form of a list of `HEAD-DEPREL` pairs (optional)
10. `MISC`: any other annotation (optional)

[3] generally a word, with some exceptions. Examples of such exceptions are Italian contractions such as *dello*, which is generally divided into *del* + *lo*.

[4] again, there are exceptions: in the case of Italian contractions and other multiword tokens, ranges may be used. Furthermore, empty nodes are characterized by decimal numbers between 0 and 1.

```
# This is a dependency tree.
1 This this PRON _ _ 5 nsubj _ _
2 is be AUX _ _ 5 cop _ _
3 a a DET _ _ 5 det _ _
4 dependency dependency NOUN _ _ 5 compound _ _
5 tree tree NOUN _ _ 0 root _ _
6 . . PUNCT _ _ 5 punct _ _
```

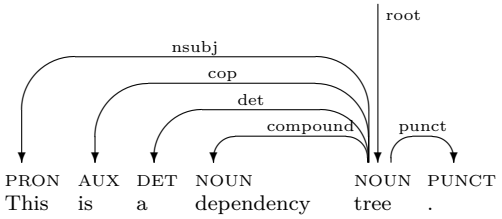


Figure 2.2: A dependency tree in CoNNL-U format alongside its graphical representation. Optional fields in the CoNLL-U file are left blank.

2.1.3.1.2 Universal POS tags

Universal POS tags mark the core part-of-speech categories, such as nouns, verbs, pronouns and determiners. An important - as we will see in Section 3.1.2.1 - distinction we can make based on POS tags is that between *content* or *open class* words, i.e. words that contribute to the meaning of the sentences they appear in (nouns, lexical verbs, adjectives, adverbs and interjections), and *function* or *closed class* words, like pronouns and determiners, so called because they do not readily accept new members[5].

While the majority of Universal POS tags correspond to the grammatical categories of traditional grammars, the UD annotation scheme does have its peculiarities. Most importantly for the following discussion, verbs are divided into lexical verbs, tagged `VERB`, and auxiliaries, tagged `AUX`. A more systematic description of `UPOS` tags is given in A.1.

2.1.3.1.3 Universal dependency relations

Universal dependency relations, largely based on [14], represent, as mentioned in the above, syntactic dependencies between individual pairs of words occurring in the same sentence. In particular, according to the CoNLL-U standard (cf. Section 2.1.3.1.1), each word is assigned a *dependency label* indicating in which way it is liked to its `HEAD`.

In this sense, the only exceptional case, which will be the starting point for our short overview of dependency relations, is that of the `root` label, generally assigned to the main verb of a sentence, ignoring any auxiliaries. For instance, *smoked* is the root of the sentence

Example 1 *"Katia has just smoked a cigarette"*

In cases such as the following, where the main verb is a copula

Example 2 *"Katia is a psychologist"*

the root is its complement and the verb is linked to it with the label `cop`, while if there is no verb at all there is no fixed rule, excepts that, in order to avoid unnecessary discrepancies between languages, the root should be a content word.

[5] a notable exception to this is the recent introduction of the gender-neutral pronoun *hen* in Swedish.

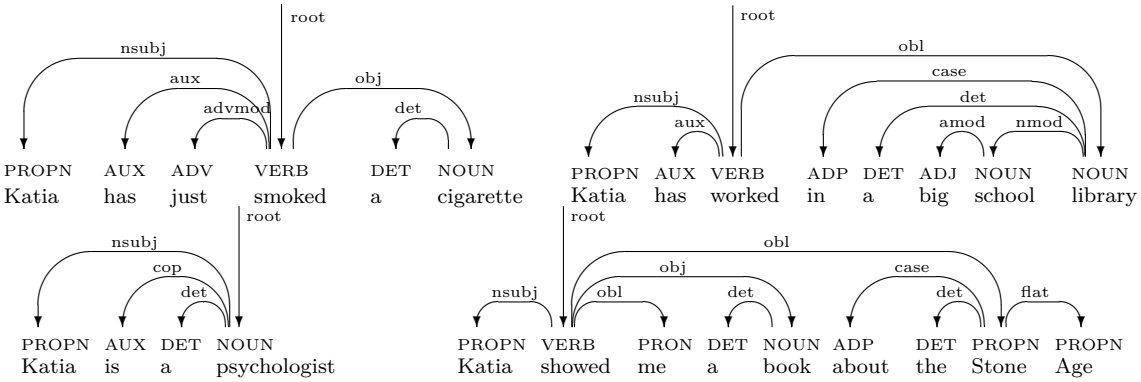
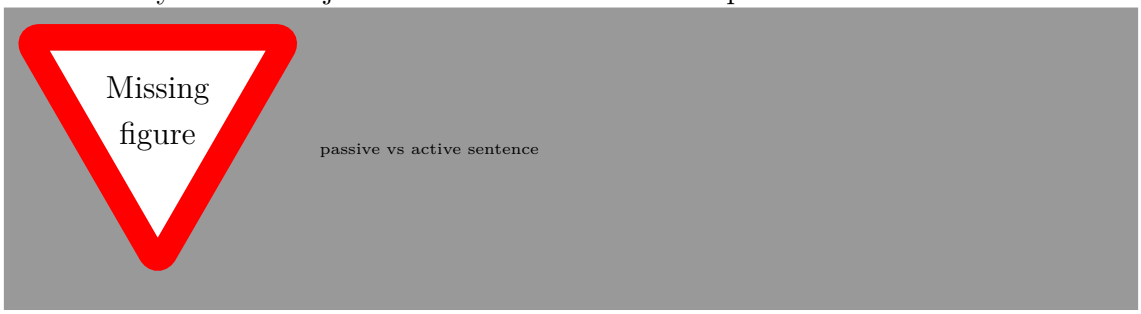


Figure 2.3: UD trees containing common dependency relations that are discussed in Chapter 3. The three on the left correspond to examples 1 and 2.

Other dependency labels commonly found in simple clauses and that will be dwelt on in Chapter 3, all exemplified in Figure 2.3, are:

- `nsubj`, marking the link between a noun, proper noun, pronoun or numeral to the `root` of a sentence or, more in general, to the head of a clause
- `aux`, marking auxiliary verbs other than the copula
- `obj`, `iobj` and `obl` marking the link between a verb and its object, indirect object and other complements respectively
- `advmod`, `amod`, `nummod` and `nmod`, marking links between modifiers and the nouns or verbs they refer to
- `flat`, indicating a flat multiword expression, and `compound`, appearing in compound nouns whenever they are written as two or more separate words.

A more detailed description of all the UD labels mentioned in this work is given in A.2. However, because the topic will arise in Chapter 3, it is also worth mentioning that CoNNL-U trees can present UD labels followed by a *subtype*, separated by the label itself by a semicolon and used to indicate grammatical relations that are specific to one language or a small group of related languages. A subtype that is commonly used in English is, for instance, `pass`, added to both the - clausal or nominal - syntactic subject and `aux` of a sentence in passive voice.



2.1.3.1.4 Relation to GF

The complementarity of constituency and dependency grammars and the multilingual nature of UD make it interesting to use in conjunction with GF, and experiments aimed at exploiting their similarity have already been performed [22, 31]. In the case of our proposed MT pipeline, dependency trees need to be at some point

converted into GF ASTs, albeit with the disadvantage that, unlike its reverse [22], the UD-to-GF conversion is a non-deterministic search problem, addressed in [31], where `gf-ud`, a program for converting UD trees into GF ASTs - and vice versa - is presented.

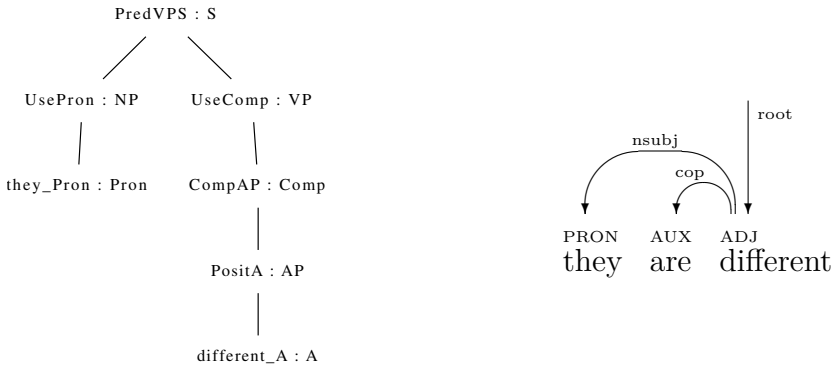


Figure 2.4: A GF AST and its UD counterpart.

2.2 Concept Alignment

In this section, we aim to give an exhaustive description of CA and of the subtasks it consists of. In order to do that, we deem it necessary to start by giving a definition of what we commonly refer to as *concepts*.

2.2.1 Concepts

Intuitively, concepts are the components of meaning, and therefore, in a multilingual context, the units of translation. If we assume the principle of compositionality to be valid and apply it to translation, these meaning components are the common denominator between an expression in the source language and its translation, which can be generated using them as the starting point.

From the compiler-like perspective described in Section 2.1.2.2, this means that concepts are what the abstract syntax should represent, i.e. that they *are* the "interlingua" of the MT system. This raises the question of how to define the abstract syntax needed by our MT pipeline starting from a corpus of parallel texts, and it is at this point that language comparison and, as such, CA, come into play.

2.2.2 Alignments

In terms of parallel text analysis, an *alignment* consists in a pair of expressions, one in the source and one in the target language, that are one the translation of the other, which means, from our compiler-like standpoint, that they share the same abstract syntax. Even though, unless otherwise specified, we will keep referring to *language pairs*, *Source Language* (SL) and *Target Language* (TL), it is easy to see how this can be generalized to a more-than-bilingual, multidirectional - in the sense that there are no set source and target language - context: the pair just mentioned simply becomes a tuple of equivalent expressions in different languages. Formally,

Definition 2 *An n-lingual alignment is an n-uple* $\langle e_1, ..., e_n \rangle$ *of semantically equivalent expressions, where each expressions* e_i *is in a different language* L_i*.*

While this general definition does not specify it, expressions are not necessarily represented as strings, but can rather be defined as tree-like structures, and in the present case as dependency trees that can later be replaced by GF concrete syntax trees. From matching concrete syntax trees to GF ASTs is a short step, as we will show in Chapter 4.

As mentioned in the introduction, the task of aligning concepts comes in two variants:

1. ***concept extraction*** (CE), which consists in identifying new concepts via mere linguistic comparison and whose output is a set of abstract syntax representations paired with the corresponding concrete expressions in the two or more languages compared
2. ***concept propagation*** (CP), i.e. the task of finding the concrete expressions corresponding to a set of known concepts in one or more particular languages not used for CE.

If, as in the case of this project, the aim is to develop a multilingual MT system, these two tasks can be seen as two potentially subsequent steps: in fact, if the first objective is that of obtaining a set of concepts by comparing two or more translations of the same text, once these concepts are in adequate number and of sufficient quality it becomes possible to provide support for additional languages by simply looking for the concrete expressions that, in the new language, correspond to each of the previously gathered concepts.

2.2.3 Approaches to automation

Some form of manual, semi- or fully automated CA is in a sense at the heart of all traditional, even very early MT systems. In the following section, we review some standard approaches from SMT. After discussing their limitations, we conclude the chapter with an overview of the grammar-based approach proposed in this thesis.

2.2.3.1 Statistical methods

In the simplest case, *word alignment* consists in finding pairs of individual words that translate to each other and store them in a dictionary. Among standard word alignment models the five IBM models [7] and their numerous variations stand out. The GIZA++ toolkit [24], an open source implementation of the IBM models based on the initial GIZA package [5], is widely used, for instance in SMT systems such as Moses [21].

As mentioned in the introduction, however, alignment can be performed at different levels of abstraction - to bring this to the extreme, we might decide to align full sentences, or even whole documents - and aligning individual words is not necessarily the best option for MT. For this reason, word alignment has been generalized to *phrase alignment*[6].

[6]it must be noted that, in the context of statistical MT, the term "phrase" is not necessarily

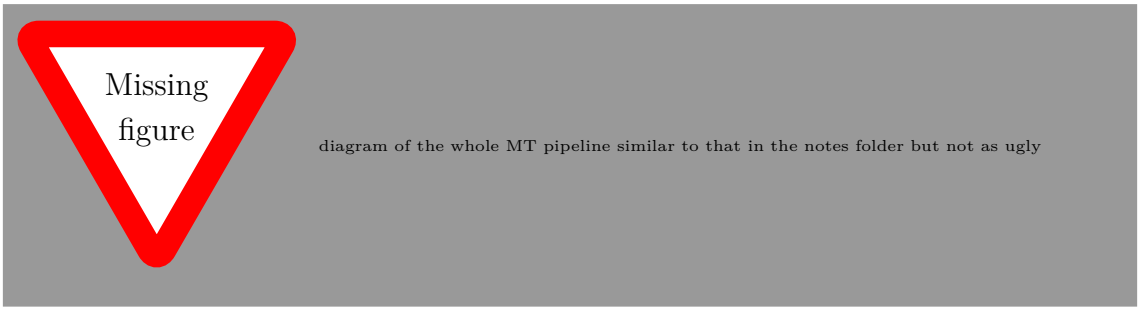
In any case, in both word and phrase alignment the relation that is established is between strings in different languages and there is no intermediate representation capturing the concepts themselves. The parallel between MT and compiler pipelines proposed in Section 2.1.2.2 suggests instead that a more flexible approach, applicable at any level of abstraction, could be grammar-based. This is particularly useful in cases, not at all uncommon, where the minimal translation units are, in one or both the source and the target language, multiword - potentially discontinuous - expressions or even more complex constructions that not even phrase alignment is able to handle.

2.2.3.2 Our approach

Given that GF provides us with a good basis for NLG and that the intermediate representation it makes use of preserves all the syntactic information present in a sentence, the idea of trying to perform CA by comparing GF ASTs seems tempting. However, since ASTs need to be obtained from raw - or, at most, barely sentence-segmented - text, CA would suffer from the same problem that, as mentioned in in Section 2.1.2.2, affects GF-based open-domain translation: the lack of a sufficiently robust analysis stage and, as a consequence, the inadequacy of the resulting parse trees.

As anticipated throughout this chapter, we attempt to solve this problem by taking advantage of dependency parsing and of the tools that have been developed with the intention of leveraging the similarities between GF and UD (cf. Section 2.1.3.1). Concretely, this means that, at a high level, the complete prototype MT pipeline we propose is composed of:

1. a UD parser
2. an alignment module based on dependency tree comparison, whose CE and CP components are described and evaluated in detail in Chapters 2 and 3 respectively
3. a program, based on `gf-ud`, that converts the alignments produced in step 2 into a domain-specific GF lexicon
4. a GF-based NLG module



We emphasize the fact that the MT pipeline presented in this thesis and, as a consequence, the GF lexicon we aim to generate are domain-specific since their purpose is not to compete with any existing MT system but rather to provide us with a way to evaluate our CA component, which, being based on syntactic

to be intended in its grammatical sense, but can refer to any sequence of - typically contiguous - words.

comparison and not, excepts marginally, data-driven, should be able to perform well - in the sense of finding many exact correspondences - on parallel corpora of any size, even on individual sentences. As a consequence, our overall evaluation, described in Chapter 4, consists in a case study conducted on only one parallel text, that of the General Data Protection Regulation (GDPR) [1], which has been chosen, building upon previous, unpublished experiments, by virtue of its public availability and of its numerous, high quality translations.

3

Concept Extraction

As mentioned in Section 2.2.2, extracting a set of concepts via linguistic comparison is the first and most important step of CA. This chapter describes our CE module, delineates our strategy to evaluate it independently from the other stages of a MT pipeline and presents the results of such preliminary evaluation.

3.1 Method and implementation

For the reasons discussed in Section 2.2.3.2, our syntax-based approach to CE is based on comparing, instead on ASTs, dependency trees, and UD trees in particular. The task of dealing with UD trees is facilitated by the existence of a series of Haskell modules written in the context of the development of the aforementioned `gf-ud` and of some preliminary unpublished CA experiments.

In this context, dependency trees are represented as a recursive data type: a tree is composed of a root node and a list of (sub)trees. Each dependency node is a record type whose fields mirror those of a CoNLL-U file.

3.1.1 Baseline

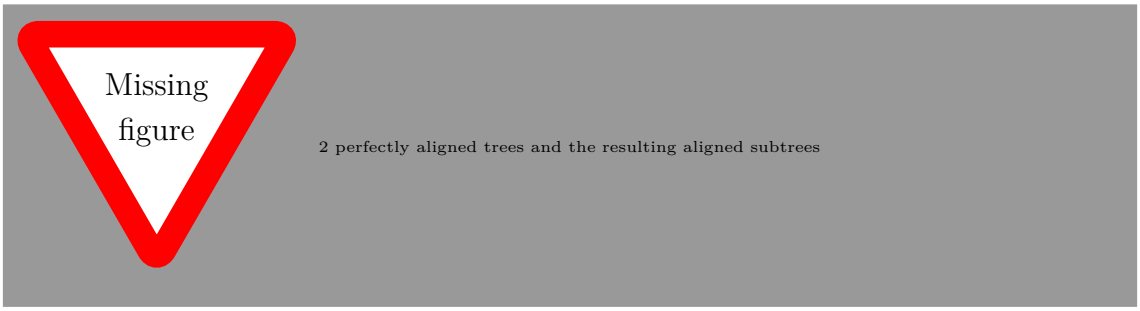
Useful both as a baseline and as a source of inspiration for our CE program, the original CE algorithm, dating back to the experiments just mentioned, consists in, given two trees corresponding to a sentence and its translation, aligning the entire trees by recursively sorting and padding their lists of subtrees. In particular:

- sorting is based firstly on the UD label of their root, and secondly on the distance of the root from its source
- what is meant by *padding* is the insertion of a dummy dependency subtree wherever a tree in the source (resp. target) language has a subtree with UD label l that is missing in its target (resp. source) language counterpart. This is useful, for instance, for dealing with cases where a determiner in a sentence in the source language is omitted in its translation.

The result is a pair of *perfectly aligned* trees, i.e. trees with identical shape that can later be used to extract the pairs of subtrees we refer to as alignments (cf. section 2.2.2). Every time a pair of subtrees is extracted, in addition, a new pair of subtrees is added for their roots. This allows to find single-word alignments that would otherwise be ignored.

pseudocode

Obviously, the "full-sentence" tree pair is itself an alignment, even though a trivial one, as it represents a sentence-level correspondence in a case where, due to the approach being syntactic comparison, the inputs are assumed to be sentences that correspond to each other[1].



2 perfectly aligned trees and the resulting aligned subtrees

3.1.2 Proposed improvements

While one of the most important ideas in the improved version of the CE module proposed in this thesis remains aligning (sub)trees with identical root UD labels, using this criterion alone makes our baseline both unable to detect many of the existing correspondences and prone to extract incorrect ones both when syntactic similarity is only apparent and when words are aligned based exclusively on the fact that they are the roots of two aligned - even with good reasons - subtrees[2]. As a consequence, the objective of this part of the project is to improve both the precision and the recall of the algorithm.

3.1.2.1 Multiple alignment criteria

One obvious way to increase the total number of alignments the algorithm detects is to, instead of considering as aligned only subtrees in *matching contexts*, i.e. whose roots are at the same depth in the sentence trees and that are attached to the same head, and with matching UD labels, make use of a set of additional, possibly more relaxed, criteria. In the following, formal definitions of all the criteria used, including the original one, are given.

[1] contrary to what it may seem, however, sentence-level alignment is in general not at a trivial task, as it is often the case, especially in certain language pairs, that one *orthographic* sentence, i.e. a sentence defined based on the presence of a full stop, maps to more than one orthographic sentences in the translated text, or vice versa.

[2] see Section 3.1.2.2.2 for a more detail discussion on head alignment.

3.1.2.1.1 Label matching

As mentioned in section 3.1.1, the only alignment criterion the original version of the CE module makes use of is based on comparing the UD labels of each pair of trees candidate for alignment, i.e. of each pair of trees in matching contexts. Formally:

Criterion 1 (Matching UD labels) *Two dependency trees t_1, t_2 could form an alignment if their roots share the same UD label.*

When referring to UD labels we disregard, unless otherwise specified, subtypes.

3.1.2.1.2 POS-equivalence

As mentioned in Section 2.1.3.1, dependency trees provide information not only on the syntactic role of each word, but also on their grammatical category, represented as a universal Part Of Speech (POS) tag. Intuitively, if the words corresponding to the nodes of two trees in matching contexts have the same POS tags, the two trees are generally more likely to correspond to each other than if not. As a consequence, a useful relation to define between dependency trees is that of *POS-equivalence*:

Criterion 2 (POS-equivalence) *Two dependency trees t_1, t_2 could form an alignment if $M_1 = M_2 \neq \emptyset$, where M_i is defined as the multiset of POS tags of all the nodes of t_i that contribute to the meaning of the corresponding sentence.*

The definition specifies that the two multisets should not contain the POS tags of all the words in the sentence but rather only those of the words *that contribute to its meaning*. What this means is that some classes of words, such as auxiliary verbs, determiners, pronouns, adpositions and conjunctions can and should be in most cases ignored as they are often omitted or rendered with words with different POS tags when the sentence is translated to another language, especially if the two languages in the pair at hand differ significantly. Words that should, in turn, generally be taken into account correspond roughly to content words (cf. Section `upos`), but these terms were deliberately avoided in Definition 2 due to the fact that it can be useful to also include some function words, for instance pronouns and some kinds of determiners. The current implementation considers as meaning-carrying all words that belong to an open class - defined as in the UD documentation [4] - and numerals, but this set of tags has been obtained empirically and might not be ideal for all language pairs.

Applied alone, this criterion can be used to capture correspondences that would otherwise be missed, thus increasing recall, but a decrease in precision is also to be expected. Perhaps more interestingly, however, as we will expand on in Section 3.2, another way to apply this criterion is in conjunction with others, and in particular together with UD label matching 1, in context where high precision is more important than recall.

3.1.2.1.3 Handling divergences

While we do not want to make our CE module language pair-specific, there are many cases where parallel texts present significant and systematic cross-linguistic distinctions. Some of these distinctions, formalized in [16], have nothing to do with idiomatic usage or aspectual, discourse, domain or word knowledge and are not

specific of particular language pairs, even though they do occur more often in some than they do in others. Drawing inspiration from [16] and [18], we refer to these distinctions as *divergences* and introduce a third alignment criterion based on them:

Criterion 3 (Known divergence) *Two dependency trees t_1, t_2 could form an alignment if they match a known divergence pattern.*

In [16], seven classes of divergences are identified: *thematic*, *promotional*, *demotional*, *structural*, *conflational*, *categorial* and *lexical*. However, because the author's proposed way to resolve divergences assumes the availability of more than merely syntactic information (in particular, of a lexicon of Root Lexical Conceptual Structures specifying, for instance, the logical subject, arguments and modifiers of each verb) and because, working with UD trees, we do not have access to anything but strictly syntactical and morphological annotations, the very task of identifying these distinctions at this level of granularity becomes, in our position, extremely challenging. Consequently, we refer to the simpler classification and less formal definitions proposed in [18], where promotional and demotional divergences are merged in a wider-coverage category of *head-swapping* divergences and where lexical divergences, the hardest to handle in the absence of any kind of semantic information, do not appear at all. In the following, we give definitions and examples for each of these classes of divergences, specifying to what extent and how[3] each of them is handled in the proposed language-agnostic CE module. We do not attempt to cover all possible cases, but rather provide a few examples as a proof of concept.

Categorial divergence A categorial divergence happens when the translation of a word with POS tag P_1 is a word with a different POS tag P_2. The instances of this class of divergences our CE module handles explicitly are some of those that cannot be captured by Criterion 1 alone due to the fact that the difference in POS tags also causes the UD labels to be different, and in particular:

- cases where an adverb in the source language corresponds to an adjective in the target language (and vice versa: all rules the program is based on are symmetrical), e.g.
  Example 3 *"Roberta listens distractedly" VS "Roberta lyssnar distraherad"*
  where the English *distractedly* is an adverb and the Swedish *distraherad* is an adjective and, as such, would be labelled respectively as an `advmod` of *lyssnar* and as an `amod` of *Roberta*
- cases where a nominal modifier in the source language is rendered as an adjectival modifier in the target language, like the following, where the English adjective *doctoral* becomes the Italian noun *dottorato*, preceded by the preposition *di*:
  Example 4 *"Herbert completed his doctoral thesis" VS "Herbert ha completato la sua tesi di dottorato"*
- cases where an adverb is rendered as an oblique, such as
  Example 5 *"Nicola studies consistently" VS "Nicola studia con costanza"*

[3] unless otherwise specified, each divergence can be expressed as a set of simple rules in the form of boolean functions taking the two UD trees candidate for alignment as input.

There are indeed other divergences of this kind that can occur in a parallel text and cannot be captured by Criterion 1, such as

Example 6 *"Claudio is hungry" VS "Claudio tiene hambre"*

where, in the original sentence, the complement of the copula *am*, *hungry*, would be labelled as its `root` but, in Spanish, the noun *hambre* is the object of the root verb *tiene*. It proved hard to detect cases like this via purely syntactic rules without causing the program to extract a lot of incorrect alignments as well, but if necessary it is easy to modify the program by removing or adding rules of this kind, potentially even language specific.

Conflational divergence Conflational divergences involve the translation of two or more words in the source language using a single word that combines their meanings in the target language. A straightforward example is that of compounds, for instance

Example 7 *"Filippo is interested in game design" VS "Filippo är intresserad av spelutvekling"*

but there are also other cases, like

Example 8 *"I'll drop by and say hello to Bruno and Andrea" VS "Passerò a salutare Bruno e Andrea"*

where the single word *salutare* is not a compound but expresses the same meaning as "say" and "hi" together. Divergences like the one described in the latter example are usually captured by Criterion 1, while compounds often need to be taken care of explicitly. For reasons that will become clear later, these cases are discussed in 3.1.2.2.2.

Structural divergence Structural divergences happen when the subject, object or indirect object of a sentence in the source language are rendered as obliques in the source language. For instance, in the sentences

Example 9 *"I called Francesco" VS "Ho telefonato a Francesco"*

what is the direct object in English (*Francesco*) becomes a prepositional phrase (*a Francesco*) in Italian. Even though in many cases these divergences can be captured by POS-equivalence, in order to ensure higher precision, the CE module handles all of them explicitly via rules that combine it with UD label checking and whose priority is higher than that of the basic Criterion 2.

Head swapping divergences Head swapping divergences always involve a head verb and a logical modifier and occur when the logical modifier is placed lower (resp. higher) in the source language sentence than in its target language counterpart. For instance, in the following example

Example 10 *"Anna usually goes for walks" VS "Anna brukar promenera"*

in Swedish, the logical modifier *usually* is implicitly expressed by the verb *brukar* itself, i.e. placed "higher up".

These divergences are, while not uncommon, hard to handle without access to a lexicon where entries for verbs are complete with a list of arguments and modifiers.

Thematic divergence Thematic divergences can happen when the logical subject of a sentence in the target language differs from its grammatical subject. An example of thematic divergence is the following:
Example 11 *"Yana likes books" VS "A Yana piacciono i libri"*
In this case, in the Italian translation, the (logical and grammatical) subject of the English sentence, *Yana*, is expressed, even though it is still the logical subject of the sentence, as an oblique; the object *books* becomes the grammatical subject (*i libri*) in Italian, while the head verb remains in both cases the root.

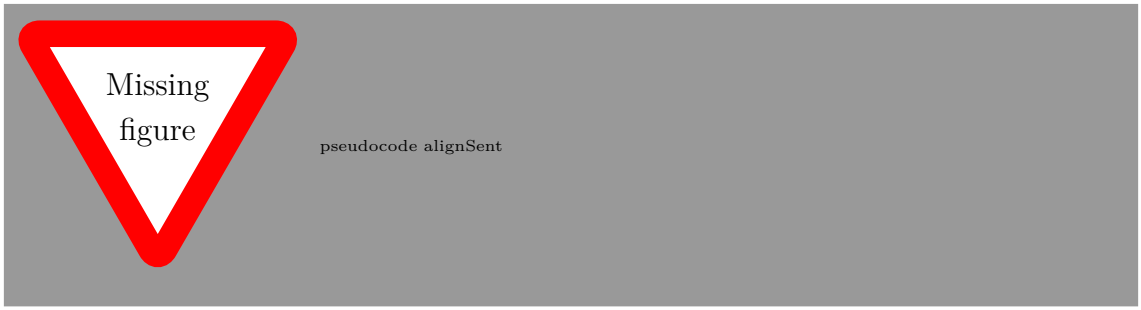
Similar to this are the cases where a passive sentence in the source language is rendered as active in the target language and, as a consequence, the subject and complements are swapped:
Example 12 *"The game had only been tried by Andrea" VS "Solo Andrea aveva provato il gioco"*

One could object that divergences like the above (and some other) could - and maybe *should* - be avoided. We will not dwell upon this, since what matters for the purpose of CA is not what is desirable in natural language translation but rather what divergences do occur in the human-translated parallel texts available for analysis.

Given that UD provides a subtype `pass` for `nsubj`s and auxiliary verbs, it is easier to explicitly handle cases such that of Example 12, while those that resemble Example 11 are left, for the moment, unhandled.

3.1.2.2 New extraction algorithm

The decision to use different alignment criteria simultaneously, together with the nature of the criteria described in the above themselves, implies that perfect alignment must be given up: the new algorithm is such that aligned subtrees are not extracted from a pair of sorted and padded sentence trees but rather obtained by direct comparison between the original trees. Aligning a pair of sentences means, then, checking if the corresponding UD trees t_1, t_2 match any of the criteria at hand and, if they do, considering them as the components of an alignment, adding the alignment to a collection and recursively repeating the same procedures for all possible pairs t'_1, t'_2 where t'_1 is a subtree of t_1 and t'_2 is a subtree of t_2.



3.1.2.2.1 Selecting the best alignments

In its most basic version, this algorithm does nothing to avoid subtrees in the source language to be aligned with multiple subtrees in the target language and vice versa.

As a consequence, it systematically *overgenerates* alignments.

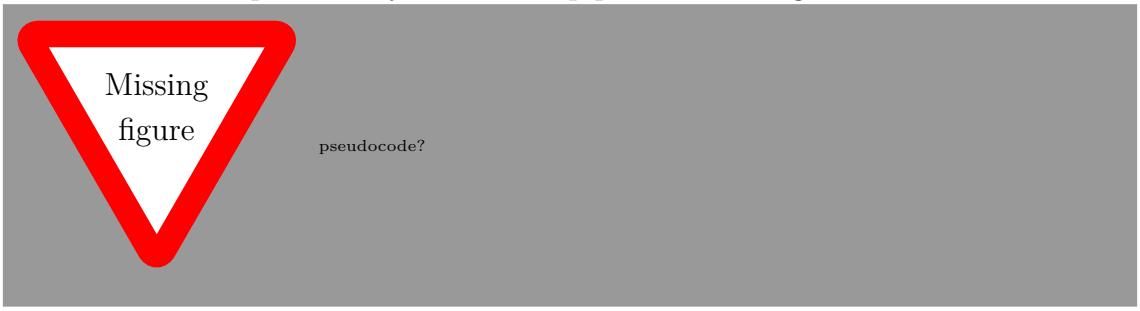
To avoid overgeneration, or rather to counter it, we need a way to select, in the very common case that the algorithm detects several alignments alternative to each other, the one that is more likely to be correct. A way to do this is to sort such alignments based on their "reliability" and only keeping the first alternative. The question becomes, then, how to sort the alignments.

To answer this, it must be first of all said that the criteria described in Section 3.1.2.1 are not to be considered equally reliable nor describe equally frequent situations. For this reason, a way to obtain a series of alignments that are, in part, implicitly sorted is to apply the criteria (or particular combinations of criteria) in a specific order that can be determined empirically and is easily modifiable thanks to the fact that, in its current implementation, the algorithm outlined above takes a list of criteria assumed to be in order of priority as one of its parameters.

It can happen, however, that two trees match *more than one* of the criteria. For instance, two trees might have their roots sharing the same UD label (cf. Criterion 1) *and* be POS-equivalent. In such cases, it intuitively makes sense to consider them more likely to be exact, as there are literally *more reasons to align them*. This makes the implicit ordering that the solution just described makes use of insufficient and leads to the necessity to keep track of the set of reasons why an alignment has been extracted - from the implementative point of view, each alignment is simply associated with a `Set` of labels, whose type is in fact named `Reason` and has an ordering defined over - and only then sort the alternative alignments by

1. the number of reasons for alignment
2. the priority level of the most important reason

These labels can even be part of the final output of the CE module, thus adding an ulterior level of explainability to the MT pipeline it belongs to.



3.1.2.2.2 Aligning heads

As mentioned in Section 3.1.1, the original version of the algorithm creates, every time an alignment is extracted, an additional one for the for the heads of the two trees in the alignment, which we shall refer to as a *head alignment*. Doing this is a crucial part of the algorithm, as the following example, concerning a straightforward to align pair of sentences, shows:

Example 13 *Consider the English sentence "Enrico eats a banana" and its Italian equivalent "Enrico mangia una banana". As the figure below shows, their trees are perfectly aligned without any need for padding or sorting:*

Without head alignments, the output of the algorithm described so far would be:

- ⟨ *Enrico eats a banana, Enrico mangia una banana* ⟩
- ⟨ *Enrico, Enrico* ⟩
- ⟨ *a banana, una banana* ⟩
- ⟨ *a, una* ⟩

while introducing head alignment leads to detecting two additional, equally relevant one-word correspondences:

- ⟨ *eats, mangia* ⟩
- ⟨ *banana, banana* ⟩

While it is of extreme importance not to miss alignments like the last two in the example above, aligning heads is not always appropriate, especially when, as in the present case, common translation divergence patterns are one of the criteria. For instance, when two trees are aligned because of a categorial divergence such as that of Example 4, where *doctoral* corresponds to *di dottorato*, it is at least questionable to also align *doctoral* with *dottorato*. Consequently, in the current implementation of the algorithm, each alignment criterion is associated with a flag telling wether heads should also be aligned whenever a new alignment is extracted because of that criterion.

In a way, even though it is handled differently from the others discussed so far, that of aligning heads could also be seen as another alignment criterion:

Criterion 4 (Heads of matching trees) *The roots n_1, tn_2 of two trees t_1, t_2 could form an alignment if t_1 and t_2 are aligned.*

There are also cases where some sort of head alignment is desirable but it is not as simple as creating an additional alignment from the roots of each pair of aligned subtrees. As a consequence, an important part of the improved CE module developed within this project is a function `alignHeads` that performs head alignment with the necessary caveats. Following are the two special cases its current implementation is able to handle correctly despite being less straightforward. As for translation divergences, there may well be other cases that can be taken care of similarly: the objective of this work is just to demonstrate how this can be done.

Auxiliaries When the translation counterpart of a verb in the SL is composed of a lexical verb and one or more auxiliaries (or vice versa), it is desirable to align the SL verb to the entire TL subtree composed of the head verb and its auxiliaries, like in the following example:

Example 14 *"Many important decisions were taken by Tommaso" VS "Många viktiga beslut togs av Tommaso"*

Achieving this is relatively straightforward: before aligning a pair of head verbs, their dependents labelled `aux` must be looked for. In case the head verb in the target language has one or more such dependents, but its source language counterpart does not, the target language component of the new alignment will not be composed by the head exclusively, but will include the `aux` subtrees.

Compounds We mentioned compounds as a type of conflational divergence (cf. Section 3.1.2.1.3). Given the variety of ways the translation equivalent of a compound can be expressed and the consequent variety of related UD labels (a compound can be rendered as another compound, as a `compound` expression, a noun modified by adjective and/or another noun, as a `flat` multiword expression, as a combination of these things...), compounds are in practice slightly harder to align than main verb + auxiliaries constructions, but the solution is conceptually the same: before aligning two heads, we check whether the list of their immediate dependents contains anything labelled `compound`, `flat`, `amod` or `nmod`. If that's the case in, for instance, *only* the sentence in the source language, then it is likely that the head of the tree in the target language is in fact a compound. If so, it is aligned not just with the head of the tree in the source language, but with the tree composed of the head and the list of subtrees labelled as `compound`, `flat`, `amod` or `nmod`. In this way, our program is generally able to find the counterparts of a compound even when they are very complex, such as in the following case:
Example 15 *"I took a course on Machine Learning techniques" VS "Jag deltog i en kurs om maskininlärningstekniker"*

3.1.2.2.3 Counting and reusing alignments

We mentioned how knowing *why* a certain alignment was identified is important to know to what extent we can trust the program to having taken the right decision. When working on a full parallel text instead of on a single sentence, another important information is the number of occurrences of that alignment (disregarding all sentence-specific information such as the linear positions of its nodes) throughout the entire corpus: unless it is the result of a systematic error (and even in this case, having it presented as a first-class alignment will help finding the mistake), an alignment occurring multiple times can be considered "more reliable" as it is in a way "corroborated". As a consequence, the return value of the main alignment function of our CE module is a Haskell `Map`, where the key is an `Alignment` - basically a pair of dependency trees - and the value is a pair whose first element is the set of reasons for that specific alignment and whose second element is its total number of occurrences.

Furthermore, the fact that an alignment has already occurred can be used as an additional, backup criterion for when none of those described in section 3.1.2.1 apply:

Criterion 5 (Known alignment) *Two dependency trees t_1, t_2 could form an alignment if such alignment belongs to a set K of known ones.*

In the case at hand, K is initialized as empty and iteratively augmented with the results of aligning each pair of sentences, but it is not hard to imagine cases in which

starting with a nonempty set would be useful.

3.2 Evaluation

We conclude this chapter by describing the data and the approaches used to evaluate the CE module described in Section 3.1 independently from the other stages of the proposed MT pipeline and by presenting the results of such early evaluation.

3.2.1 Data

Because we want the results of our evaluation of the CE module to be independent both from the previous and the successive stages of the MT pipeline outlined in Section 2.2.3.2, the data we use for this purpose is a subset of the Parallel UD (PUD) corpus, a set of multilingual treebanks in CoNLL-U format created for the CoNLL 2017 shared task on Multilingual Parsing from Raw Text to Universal Dependencies [2] by means of manual annotation[4], which prevents the CE module from failing because of parse errors.

PUD treebanks are available in 20+ languages, of which we selected, for reasons of language competences, Italian, English, Swedish and Spanish (**only the first two used in the preliminary evaluation whose results have been included in this version of the report**), and are composed of 1000 sentences taken from the news domain and from Wikipedia. Due to the lack of a gold standard to refer to in terms of CE and to the consequent need to manually assess the correctness of each alignment obtained, for most of the evaluation we only use the first 100 of these sentences.

3.2.2 Performance metrics and evaluation strategy

While precision and recall are two well-known performance metrics that would be very well suited to the evaluation of our CE module, the lack of a CE gold standard for the PUD data forced us to approximate them respectively with:

- the number of correct alignments the program is able to extract
- the ratio between such number and the total number of alignments extracted

Determining wether an alignment is correct is not, however, a completely trivial task, since there are correspondences which are correct but not easily reusable in contexts other than that in which they were found. For instance, in the following case:

Example 16 *"He missed the boat" VS "Ha perso il treno"*

it is hard to deny that *boat* has been translated as *treno*, "train", so our CE module would - and should - be able to detect it, but in most situations it is by no means desirable that such correspondence at the word level is made use of, for instance by

[4]for some languages, dependency labels were actually automatically converted to UD format from other standards. Furthermore, manual annotation often only concerns some of fields of the CoNLL-U files, but some manual annotation is generally involved in assigning POS tags and dependency relations

storing it in an bilingual lexicon! As a consequence, each alignment can be marked as:

- correct and useful for translation (`+`), at least to some extent: we observe that perfect translation equivalents that can be replaced with each other in all contexts are rare
- correct but not useful for translation (`=`). This means that the CE module is working as expected and that we are are faced with a divergence in the broader sense fo the term, i.e. potentially due to an idiomatic usage of language
- incorrect (`-`).

For instance, if we feed the CE module the pair of sentences of Example 16, as a result we get:

```
+he missed the boat|ha perso il treno[UD,POS]1
+missed|ha perso[UD,POS,HEAD]1
=the boat|il treno[UD,POS]1
=boat|treno[UD,POS,HEAD]1
+the|il[UD]1
```

Furthermore, the absence a reference solution makes it so that comparing the updated versions of the module with the baseline described in Section 3.1.1, thus focussing on measuring the improvements with respects to it rather than the quality of the results under absolute terms, is in practice the best way to assess the results. While not ideal, repeatedly performing this kind of evaluation, starting already during the early stages of its development, proved to be useful to understand the impact of each of the changes made, thus guiding further modifications and helping debugging the program when necessary.

3.2.3 Implementative details

In order to be able to perform the type of evaluation described in the above Section and because manual inspection of the results was indispensable, we made the program able to give a user-friendly output in the following format:

```
linearized SL tree|linearized TL tree[R1,R2,...,Rn]N
```

where `[R1,R2,...,Rn]` is a the list of reasons why the alignment was found, i.e. one or more of the alignment criteria described in the above and `N` is the number of occurrences of the alignment in the corpus.

With alignments in this format available, one obvious way to evaluate a specific version of the CE module is to manually assess the correctness of each of the correspondences the program identifies and compute some statistics about them. A separate Haskell module, `EvAlign`, was written to parse files in the above format and compute a series of useful statistics:

- the total number of alignments found
- the number of *distinct* alignments, i.e. the number of alignments with distinct linearizations
- the percentage of correct alignments
- the percentage of correct alignments that are also useful for translation

- the percentage of alignments found due to each combination of criteria, and how many of them are correct.

While labelling all alignments by hand was necessary to evaluate the baseline, as long as the corpus is the same it is possible to evaluate later versions of the program semi-automatically, as the correctness of many alignments is most likely already known thanks to that first round of manual annotation or to any of the successive ones. `EvAlign` can, then, also be run interactively with a labelled file and an unlabelled file as input. When this is the case, the program tries to label as many alignments as possible based on the content of the labelled file (which, of course, can even be the result of merging multiple files obtained with different versions of the program), asking the user to assess the correctness of newly found ones only. If run in this modality, the program also displayed some more statistics that make comparison it with the older, labelled file easier, such as the number of new correct alignment found and that of incorrect alignments lost.

3.2.4 Comparison with methods from Statistical MT

Another kind of evaluation that can be perfomed before the aligned dependency subtrees are converted to GF ASTs to be fed to the last stage of the pipeline is to compare them with those obtained by means of application of existing statistical algorithms. Among the well known solutions mentioned in Section 2.2.3.1, we have chosen to focus on `fast_align` [17] because of its ease of use and installation.

A Haskell script converts the CoNNL-U files that are fed to `AlignTrees` into the input format required by `fast_align`, so that the exact same tokenization is used. A second script converting its output into that used by `EvAlign` allows to reuse all of the evaluation infrastructure described in Section 3.2.2.

3.2.5 Experimental results

Table 3.1 compares the results obtained with the baseline with those obtain with the proposed improved version on the 100-sentences corpus described in Section 3.2.1.

	baseline	**improved version**
distinct alignments	1097	1190
correct (+ and =)	830 (58.12%)	958 (80.5%)
correct and useful (+)	776 (54.34%)	891 (74.87%)

Table 3.1: Comparison between the baseline and the improved version of the CE module.

As the table shows, the raw number of distinct alignments extracted increases, which suggests an increase in recall. Most importantly, the percentage of correct alignments, our approximation for precision, increases by more than 20%.

When it comes to the improved version, it also interesting to look at criterion-specific statistics, summarized in Table 3.2.

criteria	**alignments extracted**	**correct (+ and =)**
matching UD labels	39.75%	77.59%
POS-equivalence	2.86%	82.35%
known divergence	1.42%	82.35%
known alignment	0%	0%
matching UD labels + POS-equivalence	21.17%	91.66%
matching UD labels + head alignment	22.01%	67.17%
POS-equivalence + head alignment	1.26%	80%
matching UD labels + known alignment	0.25%	66.66%
POS-equivalence + known alignment	0.08%	100%
known divergence + head alignment	0.08%	100%
matching UD labels + POS-equivalence + known divergence	0.08%	100%
matching UD labels + POS equivalence + head alignment	10.92%	95.38%

Table 3.2: Criterion-specific statistics. The leftmost column specifies a criterion or combination of criteria. The central column indicates the percentage of alignments extracted because of each set of criteria, and the rightmost one specifies how many of them were marked as `+` or `-`.

One of the main things the results suggests is that, even though most alignments are still found by applying the original criterion (cf. Criterion 1), a combination of criteria 1 and 2 is still able to detect many correspondences (32.09% of the total if we combine regular and head alignments extracted in this way) while having higher (>90%) precision. This data also give an empirical confirmation of the observation made in Section 3.1.2.2.2 about the problem that the baseline has when it comes to head alignment, indicating that POS-equivalence is especially important when extracting head alignments, which are correct in 95.30% of the cases where they share the same POS tag *and* UD label, 80% of those where they only share the POS tag but only 66.66% of those where only the dependency label matches. The small size of the corpus cause Criterion 5 to remain marginal.

Finally, Table 3.3 summarizes the preliminary results of a comparison between the improved CE module and `fast_align` run with 10 iterations in EM training and symmetrizing the alignments found by running the program in both directions. Because `fast_align` is a purely statistical approach, we trained it not only on the same 100-sentence subset of PUD used for evaluating `AlignTrees` against the baseline, but also, for a fairer comparison, on the full 1000-sentences dataset. In the

latter experiment, the alignments obtained for the last 900 sentences were discarded before evaluating the results.

	`AlignTrees`	`fast_align` 100	`fast_align` 1000
extracted alignments	1190	1440	1435
correct	958 (80.5%)	410 (28.47%)	656 (45.71%)
correct and useful	891 (74.87%)	371 (25.76%)	590 (41.11%)

Table 3.3: Comparison between the improved CE module and `fast_align` trained on the first 100 only (column 2) and on the whole PUD corpus (column 3).

What could potentially make these early results misleading is that, while our CE module tries to extract correspondences at any level of abstraction ranging from single word to sull sentence, `fast_align` is only aimed at looking for alignments on the word level (even though each word in the SL can correspond to more than one word in the TL). It is entirely possible, then, then computing more relevant statistics, for instance by filtering the results obtained with `AlignTrees` so to only keep those where the depth of one or both the aligned subtrees is one, will make the total number of correct alignments found with `fast_align` bigger that that of those found with `AlignTrees`, thus making recall implicitly higher, but we do not expect too drastic changes in the percentage of correct alignments, suggesting that our system has a significantly higher precision, at least for relatively small datasets.

4

Concept Propagation

4.1 Method and implementation

4.2 Evaluation

5

Concept Alignment in Machine Translation

5.1 Method and implementation

5.2 Evaluation

6
Conclusions

6.1 Discussion

6.2 Future work

Bibliography

[1] General data protection regulation. *Official Journal of the European Union*.

[2] Multilingual parsing from raw text to universal dependencies.

[3] Universal dependencies documentation.

[4] Universal pos tags. *UD documentation*. `https://universaldependencies.org/u/pos`. Accessed: 2020-11-04.

[5] Yaser Al-Onaizan, Jan Cuřín, Michael Jahr, Kevin Knight, John D. Lafferty, I. Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical machine translation. Technical report, John Hopkins University Summer Workshop `http://www.clsp.jhu.edu/ws99/projects/mt/`, 1999.

[6] Ted Briscoe, John A Carroll, and Rebecca Watson. The second release of the rasp system. In *Proceedings of the COLING/ACL 2006 interactive presentation sessions*, pages 77–80, 2006.

[7] Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[8] Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the tenth conference on computational natural language learning (CoNLL-X)*, pages 149–164, 2006.

[9] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[10] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, pages 263–270, 2005.

[11] David Chiang and Kevin Knight. An introduction to synchronous grammars. *Tutorial available at http://www. isi. edu/ chiang/papers/synchtut. pdf*, 2006.

[12] Noam Chomsky. *Syntactic structures*. Walter de Gruyter, 1957.

[13] Haskell B Curry. Some logical aspects of grammatical structure. *Structure of language and its mathematical aspects*, 12:56–68, 1961.

[14] Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592, 2014.

[15] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454, 2006.

[16] Bonnie Dorr. Machine translation divergences: A formal description and proposed solution. *Computational linguistics*, 20(4):597–633, 1994.

[17] Chris Dyer, Victor Chahuneau, and Noah A Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, 2013.

[18] Nizar Habash and Bonnie Dorr. Handling translation divergences: Combining statistical and symbolic techniques in generation-heavy machine translation. In *Conference of the Association for Machine Translation in the Americas*, pages 84–93. Springer, 2002.

[19] Michael Johnson. Compositionality. *The Internet Encyclopedia of Philosophy.*

[20] Aravind K Joshi and Yves Schabes. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer, 1997.

[21] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

[22] Prasanth Kolachina and Aarnte Ranta. From abstract syntax to universal dependencies. In *Linguistic Issues in Language Technology, Volume 13, 2016*, 2016.

[23] Jan Landsbergen. Machine translation based on logically isomorphic montague grammars. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*, 1982.

[24] Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2000.

[25] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[26] Aarne Ranta. Grammatical framework. *Journal of Functional Programming*, 14(2):145–189, 2004.

[27] Aarne Ranta. *Grammatical framework: Programming with multilingual grammars*, volume 173. CSLI Publications, Center for the Study of Language and Information Stanford, 2011.

[28] Aarne Ranta. Explainable machine translation with interlingual trees as certificates. In *Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017)*, pages 63–78, 2017.

[29] Aarne Ranta. *Computational Grammar: An Interlingual Perspective.* Unpublished, 2020.

[30] Aarne Ranta, Krasimir Angelov, Normunds Gruzitis, and Prasanth Kolachina. Abstract syntax as interlingua: Scaling up the grammatical framework from controlled languages to robust pipelines, 2020.

[31] Aarne Ranta and Prasanth Kolachina. From universal dependencies to abstract syntax. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 107–116, 2017.

[32] Stuart Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th international conference on computational linguistics*. Association for Computational Linguistics, 1990.

[33] Milan Straka, Jan Hajic, and Jana Straková. Udpipe: trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, 2016.

[34] Zoltán Gendler Szabó. Compositionality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2020 edition, 2020.

[35] Lucien Tesnière. Éléments de syntaxe structurale. 1959.

[36] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403, 1997.

[37] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, 2001.

[38] Daniel Zeman. Reusable tagset conversion using tagset drivers. In *LREC*, volume 2008, pages 28–30, 2008.

A

Universal POS tags and dependency label

A.1 UPOS tags

A.2 DEPRELs