Syntax-Based Concept Alignment for Domain-Specific Machine Translation

Anonymous ACL submission

Abstract

1 Introduction

Grammar-based translation pipelines such as those based on Grammatical Framework (GF) (**?**) have been successfully employed in domain-specific Machine Translation (MT). What makes these systems well suited to the task is the fact that, when we constrain ourselves to a specific domain, where precision is most often more important than coverage, they can provide strong guarantees in terms of grammatical correctness.

Nevertheless, lexical exactness is, in this context, just as important as grammaticality. An important part of the design of the Controlled Natural Language (CNL) the grammar in such a system describes becomes, then, the creation of a translation lexicon. In many cases, this is done for the most part manually, resulting in a time consuming task requiring significant linguistic knowledge. When the grammar is designed based on a parallel corpus of example sentences, it is possible to automate part of this process by means of statistical word and phrase alignment techniques (**?**). None of them is, however, suitable for the common case in which only a limited number of example sentences is available.

In this paper, we propose an alternative approach to the automation of this task. While still being data-driven, our method is grammar-based and, as such, capable of extracting meaningful correspondences even from individual sentence pairs.

A further advantage of performing syntactic analysis is that we do not have to choose *a priori* whether to focus on the word or phrase level. Instead, we can simultaneously operate at different levels of abstraction, thus extracting both single- and multiword, even non-contiguous, correspondences.

For this reason, we refer to the task our system attempts to automate as *Concept Alignment* (CA).

This paper is structured as follows. Section 2 starts by giving an overview of our approach to CA, comparing it to related work, to then focus on our algorithm for extracting correspondences. It is concluded with a description of our method for converting the alignments obtained in this way to a GF translation lexicon. After that, Section 3 presents the results of our first evaluation of the system. Finally, Section 4 consists of a discussion of such results and some ideas for future work.

2 Methodology

The objective of CA is to find semantical correspondences between parts of multilingual parallel texts. We call *concepts* the abstract units of translation, composed of any number of words, identified through this process, and represent them as *alignments*, i.e. tuples of equivalent concrete expressions in different languages.

The basic use case for CA, which we refer to specifically as *Concept Extraction* (CE), is the generation of a translation lexicon from a parallel text. This can be directly compared to the numerous existing word and phrase alignment techniques.

An interesting and less studied variant of CA is *Concept Propagation* (CP), useful for cases where a set of concepts is already known and the goal is to identify the expressions corresponding to each of them in a new language, potentially even working with a different text in the same domain. While our system implements basic CP functionalities, however, in this paper we focus on CE and restrict its application to bilingual corpora.

As stated in the Introduction, most existing extraction solutions are based on statistical approaches and are, as a consequence, unsuitable for small datasets. Grammar-based approaches, making use of parallel treebanks and collectively referred to as *tree-to-tree alignment methods*, have

also been proposed (**?**), but have historically suffered from the inconsistencies between the various constituency grammar formalisms used to define grammars for the different languages and from the insufficient degree of robustness of existing parses.

This work is a new attempt in the same direction, enabled by two multilinguality-oriented grammar formalisms developed over the course of the last 25 years: Universal Dependencies (UD) (**?**) and Grammatical Framework (GF) (**?**).

While both formalisms, the former a dependency and the latter a constituency grammar, independently solve the former issue, UD is especially appealing since dependency trees are an easier target for parsing. As a consequence, several robust dependency parsers, such as (**?**) and (**?**) are available.

UD parsing alone would then be sufficient to extract (or propagate) tree-to-tree alignments, but not to automate the generation of a ready-to-use, morphologically-aware translation lexicon. This is where GF comes into play: after correspondences are inferred from a parallel text, our proposed system is able to convert them to GF grammar rules, easy to embed in a domain-specific grammar but also making it possible to immediately evaluate the system by carrying out small-scale translation experiment using pre-existing grammatical constructions implemented in GF's Resource Grammar Library (RGL) (**?**). This is made possible by `gf-ud`, a conversion tool described in (**?**) and (**?**).

Concretely, the system we propose requires the following elements, whose reciprocal relations are shown in Figure 1:

- a UD parser
- an alignment module based on dependency tree comparison
- a program, based on `gf-ud`, that converts the alignments into GF grammar rules and uses them to construct a translation lexicon

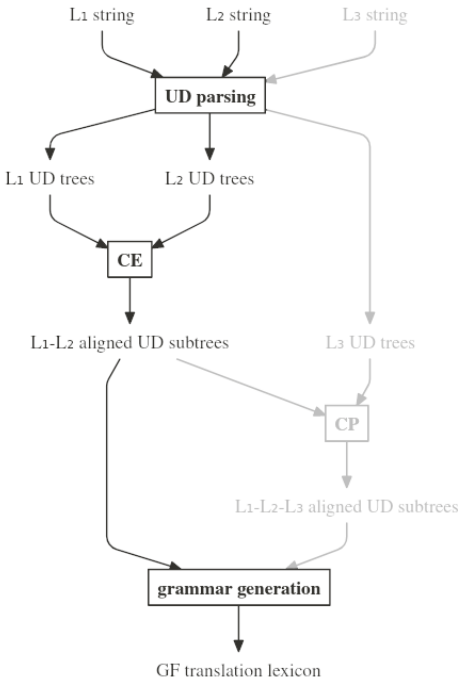

Figure 1: System overview. Parts in grey are currently at the prototype stage and will not be further discussed in this paper.

2.1 Extracting concepts

The core part of the system outlined above is of course the alignment module. In this section, we present our method for extracting alignments from parallel bilingual UD treebanks, which can be obtained from sentence-aligned texts with any UD parser.

The basic extraction algorithm, whose pseudocode is shown in Figure **??**, takes as input a list of priority-sorted *alignment criteria*, rules to determine whether two dependency trees should be aligned with each other which we will discuss in greater detail in Section 2.1.1, and a pair of UD trees corresponding to the sentences to align. From an implementation point of view, UD trees are, as in `gf-ud`, rose trees where each node represents a token, obtained from the CoNLL-U files produced by the UD parser.

As a first step, the program checks whether the two full sentence trees can be aligned with each other, i.e. if they match any of the alignment criteria. If this is the case, they are added to a collection of alignments, also represented as a pair of UD trees associated with some metadata, which is what the function will return after aligning all the dependency subtrees.

Depending on which alignment criteria they match, the head of the two trees may or may not also be added to such collection. This is important...

The same procedure is applied recursively to all pairs of immediate subtrees of each sentence, until the leaves are reached or alignment is no longer possible due to lack of matching criteria.

When working on corpora consisting of mul-

tiple sentences, this algorithm can be applied in an iterative fashion, so that knowledge gathered when a sentence pair is aligned can be reused when working on the following ones and to keep track of the number of occurrences of each alignment throughout the entire text.

2.1.1 Alignment criteria

While alignment criteria are customizable, to better understand the extraction algorithm described in the above we dedicate this section to describing the ones that are currently used by default.

Matching UD labels The most obvious, but also most effective idea is to determine alignability based on comparing the dependency labels of the candidate UD tree pair. In particular, according to this idea, two subtrees, in *matching context*, i.e. attached to aligned heads constitute an alignment if their roots share the same dependency label. Intuitively, this means that they are in the same relation with their heads.

Note that, since the root of US trees is always attached to a a fake node with an arc labelled `root`, this criterion also makes it so that full sentences are always considered to align with each other, something which is desirable since we assume the parallel texts that are fed to our program to be sentence-aligned.

Part-Of-Speech equivalence CoNNL-U files provide information not only on the syntactic role of the tokens a sentence is composed of, but also on their grammatical categories, represented as Universal Part-Of-Speech (POS) tags. Intuitively, if the nodes of two trees in matching contexts have the same POS tags, the two trees are more likely to correspond to each other than if not.

This is especially true if focus, for instance, solely on the open class words (in the current implementation, defined as in the UD documentation (**?**)), thus ignoring prepositions, determiners, auxiliary verbs and all other function words, which tend to behave differently across different languages.

As a consequence, a useful relation to define between dependency trees is that of *POS-equivalence*: two dependency trees t, u are POS-equivalent if $M_1 = M_2 \neq \emptyset$, where M_i is defined as the multi-set of POS tags of all the open class word nodes of t_i.

Applied alone, this criterion can be used to capture correspondences that would be missed by only matching UD labels, thus increasing recall, but a decrease in precision is also to be expected. However, since alignment criteria are defined as boolean functions, it is easy to combine them so to apply them simultaneously: combining these first two criteria, for instance, is useful in cases where precision is more important than recall.

Known translation divergence Parallel texts often present significant, systematic cross-linguistic grammatical distinctions. When this is the case, it is often straightforward to define alignment criteria based on recognizing the corresponding patterns. If many distinctions of this kind are specific to particular language pairs or even stylistical, some of them occur in independently of what languages are involved and have nothing to do with idiomatic usage or aspectual, discourse, domain or word knowledge. Drawing inspiration from (**?**) and (**?**), we refer to them distinctions as *translation divergences* and handle some with the alignment criteria that used by default.

Known alignment Another case in which it is trivially desirable for two subtrees in matching context to be aligned is when an equivalent alignment is already known, for instance due to a previous iteration of the extraction algorithm. When referred to pairs of alignments, the term *equivalent* indicates that the two alignments, linearized, correspond to the same string.

At a first glance, this might seem a criterion with no practical applications. However, it is particularly useful when, instead of starting with an empty set of correspondences, we initialize the program with some alignments that are either inserted manually or, most interestingly, obtained with some other alignment technique. For instance, in this way it is possible to combine the system proposed in this paper in combination with a statistical tool and give more credit to the correspondences identified by both systems.

2.1.2 Pattern matching

So far, we described how CE can be used in a setting where the objective is to generate a comprehensive translation lexicon based on set of example sentence pairs. We pointed out that the program can be configured so to prioritize precision or coverage, but never restricted our search to a particular type of alignments. Nonetheless, there can be cases in which only certain syntactic structures are of interest: for instance, for whatever reason we might be looking for adverbs or noun phrases exclusively.

To handle these situations, the CE module can filter the results based on a `gf-ud` pattern. Pattern matching in `gf-ud`

Combining pattern matching with pattern replacement, for instance by pruning the UD trees extracted by the alignment module, allows us even to extract correspondences that, given the structure of UD trees, cannot be identified by CE alone. To give an example that turned out to be useful in practice, a possibility is to extract verb phrases only, by looking for full clauses and pruning the subtree corresponding to the subject. More advanced replacements are also possible: instead of simply pruning clauses to extract the verb phrases they contain, we can perform some actual replacement to obtain *predication patterns*, such as

$$\langle XtellsYZ, XdiceZaY \rangle$$

.

2.2 Generating grammar rules

The CE module described so far outputs pairs of UD trees. While converting them to GF Abstract Syntax Trees (ASTs) is the `gf-ud`'s core functionality, to build a compilable GF translation lexicon it is necessary to turn such trees into grammar rules.

In order to do that, our grammar generation module requires a morphological dictionary of the languages at hand and an *extraction grammar*. What we refer to as an extraction grammar is of course a GF grammar, whose objective is to define the set of basic categories and syntactic rules the entries of the automatically generated lexicon is going to use.

3 Evaluation

3.1 Data

3.2 Evaluating CE

3.2.1 Experimental setup

3.2.2 Results

3.2.3 Discussion

3.3 MT experiments

3.3.1 Experimental setup

3.3.2 Results

3.3.3 Discussion

4 Conclusions

4.1 Future work