Contents

Chapter 5

Word Alignment

We now reach the core of my thesis, computing word alignments using the novel method "SimAlign" (Jalili Sabet et al. 2020) and evaluating it against two baseline methods. In the following pages, I shall give a short introduction on the topic of word alignment and explain the mechanisms behind statistical word alignment and similarity based word alignment.

5.1 Introduction

Following the success statistical models had in the task of sentence alignment, word alignment was seen as a natural extension of that work. This work had two main goals: offer a valuable resource in bilingual lexicography and develop a system for automatic translation (Brown et al. 1993).

Word alignments are objects indicating for each word in a string in the target language f which word in the source language e it arose from (Brown et al. 1993). In other words, it is a mapping of words in a string of the source language e to the words in a string of the target language f (Koehn 2009, p. 84).

A simple example for an alignment for a pair of sentences from the corpus I compiled are the German sentence *Die Beratungen sind kostenlos* "The consultations are gratuitous" and its Romansh counterpart *Las cussegliaziuns èn gratuitas*.

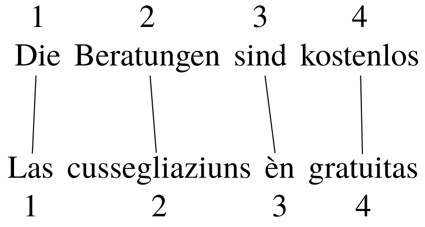
1 2 3 4
Die Beratungen sind kostenlos
Las cussegliaziuns èn gratuitas
1 2 3 4

Figure 5.1: Example of a word alignment between two sentences in German and Romansh

In this example, each word in German is aligned to exactly one word in Romansh and the words follow exactly the same order, such that the resulting alignment is the set of mappings $\{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}$. Such alignments, in which each word in the source sentence is aligned to exactly one word in the target sentence and in which the words follow the same order are considered simple (Koehn 2009, p. 85).

Things become more complicated when word order differs between languages or when several words in one sentence are mapped to one or several words in the other sentence. The latter gives rise to a variety of alignment types. A word in the target language may be aligned to several words in the source language (1-to-many alignment), or several words in the target language may be aligned to one word in the source language (many-to-1 alignment). Sometimes words in the target have no relation to the source (for instance in case of untranslatable words, or words that were omitted in the translation). In that case, they will be aligned to a special `NULL` token (Koehn 2009, p. 85).

In order to deal with these challenges of different word order and alignments that are not 1-to-1 alignments, Brown et al. 1993 developed their pipeline of translation models, the IBM Models 1-5.

5.2 Overview of Methods

I shall now give a quick explanation of word alignment methods, namely of the IBM Models, and of SimAlign, a similarity based alignment model that uses word embeddings. Since I am not a mathematician, I will not go into the mathematics of these models. I will rather attempt to explain their *modus operandi* in a more intuitive way, so as to to allow the reader some basic understanding of the mechanics behind the scenes.

5.2.1 IBM Model 1

The IBM models are translation models. They were developed in order to compute the conditional probability of a sentence in the target language f given a sentence in the source langauge e: $P(f|e)$ (Brown et al. 1993). In layman terms, they compute how likely a given sentence in the target language is a translation of a sentence in the source language. By modeling these probabilities, the models can generate a number of different translations for a sentence. However, there are infinitely many sentences in a language and most sentences occur, even in large corpora, only once. This makes the task of modeling the probability distribution for full sentences hard and not promising. Instead, the problem is broken up into smaller steps: the model models the probability distributions for individual words—it computes how likely a word in one sentence is a translation of a word in that sentence's translation. The IBM Model 1 is therefore based solely on modeling the probability distributions of lexical translations, i.e., of individual words (Koehn 2009, p. 88).

Incomplete Data

There is, however, a problem. We can compute the probability distributions of lexical translations given their counts. That is, by counting how often a word s_i^e in the sentence s_e in language e was translated as a word s_j^f in a sentence s_f in language f, we can compute the desired probability distributions. Take for example a set of German-English sentence pairs. By counting how many times the German word *das* was translated as *the*, how many times it was translated as *that*, etc., we can compute each word's translation probability distribution. With these individual probability distributions we can compute the likelihood of a sentence in language f being a translation of a sentence in language e (Koehn 2009, p. 88).

Unfortunately, while sentence alignment is a relatively easy task (at least of well-structured texts), and while sentence aligned parallel corpora are not hard to compile or come by, we do not know which words correspond to which words in the sentence pairs. This problem, dubbed as a *chicken and egg problem*, is basically the following: If we had word alignments, it wouldn't be a problem to estimate the lexical translation model and compute the probability distributions for words and sentences. And if we had a model, we could easily estimate the most likely correspondences between words in the source and the target sentences. Unfortunately, we have none of the above (Koehn 2009, p. 88).

EM Algorithm

In order to solve the problem of incomplete data, an iterative learning algorithm, the exepctation-maximization (EM) algorithm comes into play. The EM algorithm is mathematically intricate. I shall try to explain in simple words the idea behind it.

In the very first iteration, the values of the model parameters are unknown and are initialized with a uniform distribution. This means all words are equally likely to be translations of each other. Then, in the estimation step, the model is applied to the data to compute the most likely alignments. In the maximization step, the model is learned from the data based on counts collected from it. The algorithm counts co-occurrences of words in the source and the target languages, which are then weighted with the probabilities that were computed in the estimation step. These weighted counts are used to compute again the probabilities in the estimation step. These two steps, estimation and maximization, are then repeated until convergence—until a global minimum has been reached (Koehn 2009, pp. 88–92; Brown et al. 1993).

In simple words, the model does not know in the beginning which words in the source language correspond to which words in the target language. In the very first iteration, all alignments are equally likely—any word in a sentence in the target language is equally likely a translation of any word in the source language. In order to find the most probable correspondences (or alignments), the model counts how often words are aligned with each other, that is, how often they co-occur in parallel sentences (maximization step). These counts are weighted with the probabilities computed in the previous estimation step to refine the values in the next estimation step. Likely links between words are strengthened, while less likely links are weakened. This goes on until the model converges and the most likely word alignments have been learned by the model.

5.2.2 Higher IBM Models

Without going too much into details, I will shortly mention the other IBM models, Models 2-5.

Model 1 makes the unrealistic assumption that all connections for each position are equally likely. This means that word order is not modeled by Model 1. Simply put, the word order does not influence the likelihood of word alignments. Therefore, Model 2 *does* depend on word order. It adds an explicit model for alignment based on the positions of the source and the target words (Brown et al. 1993; Koehn 2009, p. 99).

Model 3 adds a probability distribution of the number of words a source word is usually translated to (dubbed *fertility*). It is able to model alignments of types other than 1-to-1 (Koehn 2009, p. 100).

Models 4 and 5 add more complexity and take into account for instance the positions of any other target words that are connected with the same source word (Brown et al. 1993), since words that are next to each other in the source sentence tend to be next to each other in the target sentence (large phrases tend to move together as units) (Koehn 2009, p. 107).

Models 1-4 serve as stepping stones towards the training of Model 5. Model 1 has a simple mathematical form and a one unique local minimum, which means the parameters learned by it do not depend on the starting point[1]. The estimates learned by Model 1 are used to initialize the training of Model 2, those of Model 2 are used to initialize Model 3, and so on and so forth—each model is initialized from the parameters of the model before it. This way, the estimates arrived at by the end of training of Model 5 do not depend on the initial estimates of the parameters for Model 1 (Brown et al. 1993).

These models have been playing a key role in word alignment tasks and in statistical and phrase based machine translation. Put together in a pipeline of models, they serve as the groundwork for Giza++, a toolkit for training word-based translation models. Using these alignments, phrase alignments can be learned in order to train a statistical phrase-based machine translation (Och and Ney 2000; Koehn, Och, and Marcu 2003)

[1]The other models have several minima; this means according to the starting parameters, different minima can be arrived at.

5.3 Word Embeddings

A different approach to word alignment is based on word embeddings. But what are word embeddings?

5.3.1 Excursion: Words

Before we discuss word embeddings, I would like to write a few words about words and their meanings.

Words are actually an arbitrary way to split linguistic material into units. What we refer to as words are usually units separated by a whitespace in writing, but the use of whitespaces is arbitrary and inconsistent. There is no real phonetic motivation for splitting units into words. Some single words sound exactly like two other words (*a maze* sounds like *amaze* and *in sight* like *incite*). The words *someone* and *anyone* are written as one word, while *no one* is written as two words, although there is obviously no difference in character between them (Jespersen 1924, pp. 92–95).

For the sake of simplicity, I will stick to the term *word*, referring to any linguistic unit, made up of one or several morphemes (or words), divided in written form by whitespaces from its neighboring units.

Meaning of Words

The question of describing the meanings of words is an entire field—semantics. But already in his posthumously published work *Cours de linguistique générale* ("Course in General Linguistics") from 1916, the Swiss linguist and semiotician Fredinand de Sassure came to an important conclusion. Linguistic elements receive their value only by being arranged in a sequence, which de Saussure calls *syntagm*: "A term in the syntagm acquires its value only because it stands in opposition to everything that precedes or follows it, or to both." (Saussure 1959, p. 123). Further, each term in the syntagm, in the sequence of terms, has associative (or paradigmatic) relations. These relations reside in the memory of the speakers. For instance the German word *zudrehen* "close something by turning" unconciously calls to mind related words, such as other words beginning with *zu-*: *zumachen* "close", *zumauern* "wall something up", *zuklappen* "close something shut". But also words with the verb *drehen*: *aufdrehen* "turn open", *verdrehen* "twist, contort", etc. etc. (Saussure 1959, pp. 122–127) (my examples).

Each term in the syntagm stands in opposition not only to the preceding and following parts in the syntagm, but also to terms in the paradigm, which are called to mind by the associative series. The meaning, or rather value of words, is a result of an intersection of two axes—the syntagmatic, the horizontal axis, and the paradigmatic one, the vertical axis.

Take, for instance, the sentence *I am drinking coffee*. The word *coffee* gets its **syntagmatic** value from the perceding word *drinking*, which stands in **paradigmatic** opposition to other words (*plant*, *grow*) which would give *coffee* a different meaning. We know that by *coffee* a hot-drink is meant, because it follows the verb *drink*. In the sentence *I grow coffee* it would mean a plant or a tree, in *I bought one pound of coffee* it would mean beans and in *coffee ice-cream* it would describe a flavor.

The Austrian-British philosopher, Ludwig Wittgenstein, summed the meaning of the word *meaning* (German *Bedeutung*) in two sentences in his Philosophical Investigations, no. 43:

> *Man kann für eine große Klasse von Fällen der Benützung des Wortes »Bedeutung« – wenn auch nicht für alle Fälle seiner Benützung – dieses Wort so erklären: Die Bedeutung eines Wortes ist sein Gebrauch in der Sprache.*[2] [3]

5.3.2 Word Embeddings

These ideas, which were further developed by linguists in the 1950's, that a word can be defined by its environment or distribution, i.e., by its set of contexts in which it occurs and its grammatical environments, is the inspriation for what is called vector semantics. The idea of vector semantics is to represent a word as a point in some n-dimensional vector space. These vectors are called **embeddings**. There are different ways and versions of word embeddings, but in each case the values of the vectors are based in some way on counts of neighboring words (Jurafsky and Martin 2019, pp. 98–99).

One version of word embeddings comes from neural language models. Language modeling is the task of assigning probabilities to a sequence of words, that is, modeling how likely it is that a sequence of words in a language would be uttered/written by a spaker of that language (Koehn 2009, p. 181). In practice, the task of a language model is predicting upcoming words from prior word context (Jurafsky and Martin 2019, p. 137).

Without going too much into details, a neural network is a complex non-linear function. It is made up of layers, which are vectors, and weights, which are matrices. The numbers (a vector) from each layer are passed on to the next layer by means of matrix multiplication with the weights between those layers. The vector resulting from this matrix multiplication (plus usually some non-linear activation function), is the next layer in the neural net. The output of a neural network can be for instance a single value, as in the cases of a binary classification task, in which the output is either 0 or 1, but it can also be a vector representing some probability distribution.

[2]For a large class of cases of the use of the word *meaning*—and maybe for all of its use cases—one could explain the word as follows: The meaning of a word is its use in the language.

[3]`https://www.wittgensteinproject.org/w/index.php?title=Philosophische_Untersuchungen#43`

In the course of the training of a language neural model, i.e., while the neural net learns the probability distributions for words given its neighboring words, the parameters for the weights are learned. The weights connecting the input layer with the first hidden layer are our said word embeddings. When inputting a word into the net (in form of a one-hot vector), we can get its vector representation, i.e., its embedding, from the so-called embedding layer. Since this representation is conditioned on context, similar words should have similar embeddings (Koehn 2020, pp. 104–105).

There are different ways of learning word embeddings. One of the most popular methods are *word2vec* (actually made up of two different methods) and *GloVE*. These methods are simpler than neural language models (Jurafsky and Martin 2019, p. 111); their main goal is to learn high quality word vector representations, not to generate language.

5.3.3 Word Similarity

If words are represented by vectors, we need a measure for taking two such vectors and measuring how similar they are. The most common similarity metric is the **cosine similarity**—measuring the angle between the vectors.

Again, without going into too much mathematical details, using the dot product for measuring similarity, i.e., multiplying the vectors with each other, favors long vectors, that is, vectors with high values in each dimension, which represents the frequency of words. This means more frequent words have higher values, but we want to measure the similarity between words regardless of their frequency. To solve this problem, we need to normalize the dot product by dividing it by the lengths of the vectors. Thus, the cosine similarity metric between two vectors **v** and **w** can be computed as:

$$\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{5.1}$$

with $\sum_{i=1}^{N} v_i w_i$ being the dot product of the vectors **v** and **w** and $\sqrt{\sum_{i=1}^{N} v_i^2}$ being the length of the vector **v**.

(Jurafsky and Martin 2019, pp. 103–104)

The cosine similarity returns a value between -1 and 1. The highest similarity is 1: the vectors are parallel and pointing in the same direction. If it is 0, the angle between the vectors is a 90° angle. The lowest similarity is -1: the vectors point in opposite directions.

5.3.4 Multilingual Word Embeddings

There are also methods for computing multilingual word embeddings. Multilingual word embeddings are word embeddings for words in different languages that share the same vector space. This can be achieved by learning word embeddings for each language separately on monolingual data, and then map these embeddings to a shared vector space (Artetxe, Labaka, and Agirre 2018). They can also be extracted from a multilingual language model (Jalili Sabet et al. 2020).

The idea behind multilingual word embeddings is that two equivalent words in different languages should have a similar distribution, thus their vector representations should also be similar (Artetxe, Labaka, and Agirre 2018).

5.3.5 Summary

Word embeddings are vector representations of words learned by a neural language model or by a more simple embeddings model. These vectors' dimensions usually range between 100 and 1000 dimensions. Similar words (words that appear in the same context) have similar word embeddings. To measure word similarity, we measure the similarity between their embeddings using the cosine similarity. Multilingual word embeddings are word embeddings for words in different languages which share the same vector space. Similar words of different languages should have similar embeddings.

5.4 Similarity Based Word Alignment

With multilingual word embeddings, similar words in different languages should have similar embeddings. This similarity between embeddings in different languages can be leveraged in order to find word alignments using a similarity matrix, without the need for parallel data, which is the idea that forms the basis of SimAlign (Jalili Sabet et al. 2020).

5.4.1 Method

SimAlign takes two parallel sentences s^e and s^f of lengths l_e and l_f in languages e and f. For this sentence pair a *similarity matrix* is defined as $S \in [0,1]^{l_e \times l_f}$. It is a matrix the size of the lengths of sentences. Each cell in the matrix will be filled with a value between 0 and 1, returned from a function measuring similarity between the embeddings of two words. This means that for each combination of two words from sentence s_e and sentence s_f, their similarity measure is filled into the corresponding cell in the matrix (Figure 5.2). From this similarity matrix S, a binary alignment matrix $A \in \{0,1\}^{l_e \times l_f}$ is extracted. The cell A_{ij} in the alignment matrix A will be filled with 1 (which means i and j will be aligned) if the word s^e_i in the sentence s^e is the most similar to the word s^f_j in the sentence s^f and vice versa (Figure 5.3).

		1	2	3	4
		Ich	liebe	ja	Äpfel
1	I	0.9	0.2	0	0.2
2	love	0.1	0.9	0	0.1
3	apples	0.1	0.1	0	0.9

Figure 5.2: Similarity matrix $S \in [0,1]^{l_e \times l_x}$, filled with values between 0 and 1 corresponding to the similarity measure between the embeddings of the words. The numbers are fictive.

		1	2	3	4
		Ich	liebe	ja	Äpfel
1	I	1	0	0	0
2	love	0	1	0	0
3	apples	0	0	0	1

Figure 5.3: Alignment matrix $A \in \{0,1\}^{l_e \times l_f}$ extracted from the similarity matrix S. The two most similar words in row i and column j of S will receive a score of 1; the rest 0.

That is, a cell A_{ij} in the matrix A is set to 1 if:

$$(i = \arg\max_{l} S_{l,j}) \wedge (j = \arg\max_{l} S_{i,l})$$

If all entries in a row i or a column j of S are 0 (as is the case in column 3 of Figure 5.2), A_{ij} will be set to 0. The resulting alignment can be seen in Figure 5.4.

This basic method is referred to in Jalili Sabet et al. 2020 as **Argmax**. Mutual argmaxes can be rare, which is why for many sentences Argmax only identifies few alignments. To remedey this, Argmax is applied iteratively in a method called **Itermax**. In each iteration, the model focuses on still unaligned pairs and tries to align them. If the similarity with an already aligned word is very high, the model can add another alignment edge. This allows for one word to be aligned to multiple other words, i.e., create 1-to-many alignment.

Argmax finds a local optimum and Itermax is a greedy algorithm. There is a third alignemnt method, called **Match**, which finds global optima. The alignments generated with the Match method are inherently bidirectional (the source is aligned to the target and

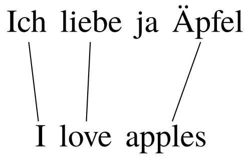

Figure 5.4: The resulting word alignment

the target is aligned to the source).

For the task for word alignment, SimAlign can use multilingual embeddings which were learned in advanced from monolingual data and then mapped to a shared vector spaced. SimAlign can also use out-of-the-box the embeddings from two multilingual language models: mBERT, which is a version of BERT (Devlin et al. 2018) trained on 104 languages[4], and XLM-RoBERTa base, trained on 100 languages (Conneau et al. 2020).

5.4.2 Summary

By measuring the similarity between multilingual word embeddings, word alignments for sentence pairs can be computed.

Multilingual embeddings can be learned from monolingual data, and thus word alignment can be computed even in low-resource scenarios, i.e., in scenarios where parallel data is scarce, which makes similarity based word alignment a competitive method against statistical methods.

Traditional statistical methods such as the IBM Models (Brown et al. 1993) and their implementations, such as GIZA++ (Koehn, Och, and Marcu 2003) or fast_align (Dyer, Chahuneau, and Smith 2013) require a large amount of data to perform well. The quality of the alignments deteriorates quickly when the size of data diminishes[5].

.

Similarity based word alignment using embeddings extracted from mBERT or XLM-R outperforms any state-of-the-art statistical method for the language pairs Czech, German, French and Hindi, paired with English. But all of these languages are part of mBERT's and XLM-R's training data. Jalili Sabet et al. 2020 emphasize the advantage of their method performing well also on a small amount of parallel data. But will SimAlign perform just as well for data unseen by said language models?

[4]`https://github.com/google-research/bert/blob/master/multilingual.md`

[5]In Och and Ney 2000, the average error rate (AER) for aligning 1.5M sentence pairs is 9.4%. When aligning only 50,000 sentences, the AER goes up to 15.6% (see Table 4 in Och and Ney 2000)

Glossary

Graubünden	The Canton of Grisons. 1, 5, 9
Standeskanzlei	State Chancellery of Grisons. 9

Acronyms

AER	average error rate. 40, 49, 52, 53, 54, 62
EM	exepctation-maximization. 33
NER	named entity recognition. 2
NMT	neural machine translation. 26, 29
POS	part of speech. 2
SMT	statistical machine translation. 25

List of Tables

List of Figures

List of Listings

Bibliography

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre (July 2018). “A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 789–798. DOI: `10.18653/v1/P18-1073`. URL: `https://aclanthology.org/P18-1073`.

Brown, Peter F. et al. (1993). “The Mathematics of Statistical Machine Translation: Parameter Estimation”. In: *Computational Linguistics* 19.2, pp. 263–311. URL: `https://aclanthology.org/J93-2003`.

Conneau, Alexis et al. (July 2020). “Unsupervised Cross-lingual Representation Learning at Scale”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 8440–8451. DOI: `10.18653/v1/2020.acl-main.747`. URL: `https://aclanthology.org/2020.acl-main.747`.

Devlin, Jacob et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: `10.48550/ARXIV.1810.04805`. URL: `https://arxiv.org/abs/1810.04805`.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith (June 2013). “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*. Atlanta, Georgia: Association for Computational Linguistics, pp. 644–648. URL: `https://aclanthology.org/N13-1073`.

Jalili Sabet, Masoud et al. (Nov. 2020). “SimAlign: High Quality Word Alignments Without Parallel Training Data Using Static and Contextualized Embeddings”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 1627–1643. DOI: `10.18653/v1/2020.findings-emnlp.147`. URL: `https://aclanthology.org/2020.findings-emnlp.147`.

Jespersen, Otto (1924). *The Philosophy of Grammar*. 1965th ed. New York: W.W. Norton & Company Inc. ISBN: 978-0-393-00307-9.

Jurafsky, Daniel and James H. Martin (2019). *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech*

Recognition. Third Edition Draft. URL: https://web.stanford.edu/~jurafsky/slp3/.

Koehn, Philipp (2009). *Statistical Machine Translation*. Cambridge University Press.

— (2020). *Neural Machine Translation*. Cambridge University Press. DOI: 10.1017/9781108608480.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu (2003). "Statistical Phrase-Based Translation". In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 48–54. DOI: 10.3115/1073445.1073462. URL: https://doi.org/10.3115/1073445.1073462.

Och, Franz Josef and Hermann Ney (Oct. 2000). "Improved Statistical Alignment Models". In: *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. Hong Kong: Association for Computational Linguistics, pp. 440–447. DOI: 10.3115/1075218.1075274. URL: https://aclanthology.org/P00-1056.

Saussure, Ferdinand de (1959). *Course in General Linguistics*. Ed. by Charles Bally and Albert Sechehaye. Trans. by Wade Baskin. New York: Philosophical Library. URL: https://ia902704.us.archive.org/35/items/courseingenerall00saus/courseingenerall00saus.pdf.