Contents

Chapter 5

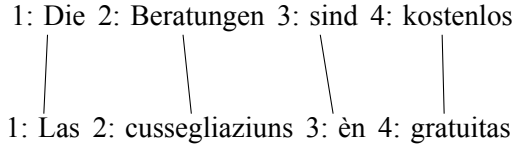
Word Alignment

We now reach the core of thesis, computing word alignments using the novel method SimAlign suggested by Jalili Sabet et al. 2020 and evaluating it against a baseline method.

5.1 Introduction

Following the success statistical models had in the task of sentecne alignemnt, 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*.



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 exactly one word in the target sentence in which the words follow the same order are considered simple (Koehn 2009, p. 85).

Things become more compliacted when word order differs between languages or when several words in one sentence are mapped to one or several words in the other sentence. A word in the target language may be aligned to several words in the source language (1-to-many), or several words in the target language may be aligned to one word in the source language (many-to-1). Sometimes words in the target have no realtion to the source (for instance in case of untranslatable words, but

words might also generally be 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 review of word alignment methods, that is of the IBM Models, of `fastalign`, which is an improved version of the IBM Model 2 and serves as a baseline model in my experiment, and of `SimAlign`, the model based on the novel method of 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 principle of operation in a more intuitive way 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 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 disributions 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 w_e in language e was translated as a word w_f in language f, we can compute the desired probability distributions. For example, 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 given a sentence in language e.

Unfortunately, while sentence alignment is a realtively easy task (at least for 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 correspon-

dences between words in the source and the target languages. 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 expectation-maximization algorithm (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 unkown 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-occurences 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 is reached and the probabilities computed stop changing.

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 weekened. This goes on until the model converges and the most likely word alignments have been learned by the model (Koehn 2009, pp. 88–92; Brown et al. 1993).

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, word order is not modeled by Model 1. In other words, the word order does not influence the likelihood of word alignments. Therefore, for 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 intialize the training of Model 2, those of Model 2 are used to initialize Model 3, and so on and so forth—each model is initiailized 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 stastitical and phrase based machine translation. Put together in a pipeline of models, they serve as the groundwork for Giza++, a toolkit for training word-basewd 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)

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 meaning.

Words are actually an arbitrary way to split linguistic material into units. What we refer to as words are usually units seperated by a whitespace in writing, but the use of whitespaces is arbitrary and incosistent. Some words sound exactly like two other words (*a maze* sounds like *amaze* and *in sight* like *incite*). *someone* and *anyone* are written as one word, while *no one* is written as two words (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 writing 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 everyhting that percedes or follows it, or to both.” (Saussure 1959, p. 123). Further, each term in the syntagm, in the sequence of terms, has associative relations. These relations reside in the memory of the speakers. For instance the German word *zudrehen* “close something

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

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).

Each term in the syntagm stands in opposition not only to the perceding and following parts in the syntagm, but also to terms 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.

In 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

[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`

for instance a single value, for example in cases of a binary classification task, the output is either 0 or 1, or it can be a vector representing some probability distribution.

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 good word vector representations, not generating 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., multipliying the vectors with each other, favors long vectors, that is vectors with high values in each dimension. 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:

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

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

5.3.4 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.

5.4 Embedding Based Word Alignment

5.4.1 SimAlign

5.5 Computing the Word Alignments

Chapter 7

Evaluation

Chapter 8

Contributions

Chapter 9

Summary

Bibliography

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.

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.