

Pivot-based word alignment

Tom McCoy
Advised by Robert Frank

April 28, 2017

Submitted to the faculty of the Department of Linguistics
in partial fulfillment of the requirements for the degree
of Bachelor of Arts

Abstract

Word alignment is the task of, given two sentences that are translations of each other, determining which words correspond to each other across the two sentences. Word alignment is an important step in the pipeline of constructing a statistical machine translation system, but success at word alignment depends heavily on the quantity of training data available. The traditional methods for computational word alignment, proposed by Brown et al. (1993), require large quantities of training data. However, these methods fall short when such quantities of data are not available. To combat this problem, I propose a framework that fills in the data gap by using data from languages related to the one for which data are lacking. This technique is shown to improve significantly upon the baseline alignment error rate.

In my proposed framework, aligning a sentence in a low-resource language with a sentence in a high-resource language follows a 3-step procedure. First, a pivot language is chosen such that it is both high-resource and as closely related to the low-resource language as possible. Edit distance is then used to create a correspondence between the low-resource language and the pivot language, while probabilities trained using the IBM Models are used to create a correspondence between the pivot language and the high-resource language. I test several different settings for these three basic components on the task of aligning Spanish and English, and I find that the most successful overall alignment system uses Portuguese as the pivot language, a cognate-based algorithm for calculating edit distance, and translation, distortion, and alignment probabilities from the IBM Models. With this framework settled on, I then conduct some sample translations to demonstrate the utility of this approach for machine translation. These translations are translating from Spanish to English, and despite being trained only on Portuguese the translator still manages to yield rough translations of Spanish text.

Contents

Abstract	i
1 Introduction	1
2 Word alignment	3
2.1 Previous approaches	3
2.1.1 The IBM Models	3
2.1.2 HMM word alignment	7
2.1.3 Phrase-based machine translation	8
2.1.4 Neural machine translation	8
2.2 The problem: alignment without training data	9
3 Proposal: Pivot-based word alignment	10
3.1 Outline	10
3.2 Example	11
4 Experiments	12
4.1 The languages	12
4.2 The corpus	12
4.2.1 Training sets	13
4.2.2 Test set	13
4.3 Measuring success	14
4.3.1 Precision	14
4.3.2 Recall	15
4.3.3 Alignment error rate	15
4.4 Baselines	15
4.4.1 Lower-bound baselines	15
4.4.2 Upper-bound baselines	16
4.4.3 Baseline results	16
4.5 Edit distance experiments	17
4.5.1 Levenshtein edit distance	17
4.5.2 Feature-based edit distance	19
4.5.3 Embedding-based edit distance	20

4.5.4	Cognate-based edit distance	24
4.5.5	Edit distance results	28
4.6	Word alignment experiments	34
4.6.1	Translation probability	34
4.6.2	Alignment probability	36
4.6.3	Fertility	37
4.6.4	Distortion	38
4.6.5	Word alignment results	39
4.7	Pivot language experiments	40
5	Summary of results	45
5.1	Results	45
5.2	Example translations	45
6	Conclusion	50

Chapter 1

Introduction

In the field of natural language processing, success of an algorithm often depends upon access to very large quantities of data upon which to train the algorithm; as the saying goes, there's no data like more data! Good data sets are at such a premium that some of the most-cited works in the field of NLP are the releases of data sets, such as the Europarl corpus (Koehn 2005) with 2,138 citations, and the Penn Treebank (Marcus et al. 1993) with 6,355 citations. For mainstream languages such as English and French, this requirement of big data is not a big problem because there are large corpora available for such languages. However, for several thousand other languages, there is nowhere near enough training data available to enable the creation of high-quality NLP resources. Such languages are termed "low-resource languages."

One task that clearly demonstrates the problem with data shortages is word alignment. Word alignment is the task of, given two sentences that are translations of each other, determining which words correspond to each other semantically across the two languages. For example, for the Kinyarwanda sentence and its English translation in (1a) from Gerdtz and Whaley (1991), the word alignment between the two sentences is represented by (1b):

- (1) a. Ikárámu umukoóbwa a-ra-andik-a íbárúwa nziza ná yo.
pen girl she-PRES-write-ASP letter good with it
'The pen, the girl is writing a nice letter with it.'
- b. Ikárámu umukoóbwa araandika íbárúwa nziza ná yo
The pen, the girl is writing a nice letter with it
-

Word alignment is an important step in many pathways toward building a machine translation system, such as in the popular Moses software system (Koehn et al. 2007), and effective word alignment depends heavily upon the size of the corpus upon which the word alignment algorithm is trained. Therefore, success at word alignment (and, by extension, machine translation programs based on word alignment) suffers greatly from a data shortage.

In this paper, I suggest a method to overcome data shortages for the purpose of word alignment. Specifically, in the circumstance where one wishes to align sentences between two languages, one of

which is low-resource and one of which is not, I propose making use of a high-resource pivot language closely related to the low-resource language to compensate for the lack of data in the low-resource language. For example, consider again the two sentences in (1b), but suppose there is no background information available on Kinyarwanda (the language in which the top sentence is written). I hypothesize that the computer can compensate for this lack of Kinyarwanda data by incorporating data on Swahili, which is related to Kinyarwanda since both are Bantu languages and which does have far more digital data than Kinyarwanda. For example, by noting that the Kinyarwanda word *ibárúwa* resembles the Swahili word *barua*, and armed with the background knowledge that *barua* is the Swahili word for *letter*, the computer can then make the reasonable assumption that *ibárúwa* aligns with *letter* because words that appear similar in two related languages are likely to be cognates and to thus have a significant degree of semantic similarity.

My approach utilizes edit distance to determine likely cognates in the pivot language for the unknown words in the low-resource language, while I compute parameters for aligning pivot language words with words in the high-resource language by using modified versions of the IBM word alignment models. Thus, there are three basic components to my model—a pivot language, an edit distance algorithm, and an alignment model—and I will test several different options for implementing each of these.

Chapter 2

Word alignment

2.1 Previous approaches

This section details previous approaches to the task of word alignment and to machine translation (the larger task which is the main purpose for conducting word alignment).

2.1.1 The IBM Models

Probably the most influential method for word alignment is the series of IBM models created by Brown et al. (1993). These models make use of four quantities, namely translation probability, alignment probability, fertility, and distortion probability, defined here:

Translation probability: For an English word e and a Portuguese word p , the translation probability $t(e|p)$ is the probability that e will be the translation for p .

Alignment probability: The alignment probability $a(j|i, l, m)$ is the probability that, given a Portuguese sentence of length l words and an English sentence of length m words, the i^{th} Portuguese word will be aligned with the j^{th} English word.

Fertility: For a given Portuguese word p , its fertility $n(s)$ is the number of English words that are aligned with p . For each Portuguese word s , there is a fertility distribution describing how likely it is that p will have a fertility of 1, or of 2, or of 3, etc.

Distortion probability: With j_{p_k} defined as the index within the English sentence to which Portuguese word p_k is aligned, the distortion probability $d(j_{p_i} - j_{p_{i-1}})$ is the probability of having a distance of $j_{p_i} - j_{p_{i-1}}$ between the English words aligned to two consecutive Portuguese words.

Each of these quantities makes intuitive sense, but it is not particularly clear how to actually quantify them. For example, even though English *big* and Portuguese *grande* correspond roughly in meaning, it is not clear exactly what the value of $t(\text{big}|\text{grande})$ should be, nor is it clear how the value of $t(\text{big}|\text{grande})$ should compare with $t(\text{huge}|\text{grande})$ or $t(\text{large}|\text{grande})$. The IBM models

provide a way to quantify these abstract probabilities so that they can be used for actual NLP applications.

There are five different IBM models of word alignment, each making use of different subsets of the above four quantities. In general, each subsequent IBM model builds upon the previous one. These IBM models are described in the following subsections. Each model operates under similar principles—namely, using an expectation-maximization algorithm to find the most likely values of the parameters under concern—but what varies are the sets of parameters being maximized over.

IBM Model 1

IBM Model 1 chooses a word alignment based solely on translation probability. For each pair of a Portuguese word p and an English word e , the model will establish $t(e|p)$, which is the translation probability of e being the word that corresponds to the given Portuguese word p . To determine these values, every possible $t(e|p)$ for every possible pair of a Portuguese word p and an English word e is initialized as

$$t(e|p) = \frac{1}{|\text{English lexicon}|} \quad (2.1)$$

Then, an expectation-maximization (EM) algorithm is run to tune these translation probabilities. At each step, information about which English and Portuguese words occur together in aligned sentences is combined with the translation probabilities already determined to yield new translation probabilities. The exact equation used to update the translation probabilities is as follows:

$$t(e|p) = \frac{\sum_{(\mathbf{e}, \mathbf{p})} c(e|p : \mathbf{e}, \mathbf{p})}{\sum_e \sum_{(\mathbf{e}, \mathbf{p})} c(e|p : \mathbf{e}, \mathbf{p})} \quad (2.2)$$

where $c(e|p : \mathbf{e}, \mathbf{p})$ is a function that, for a given pair of an English sentence \mathbf{e} and a Portuguese sentence \mathbf{p} , incorporates both how often the words p and e co-occur as well as how likely they are to correspond under the current model:

$$c(e|p : \mathbf{e}, \mathbf{p}) = \frac{t(e|p)}{\sum_{i=0}^{l_e} t(e|p_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_p} \delta(p, p_i) \quad (2.3)$$

$$\delta(a, b) = \begin{cases} 1, & \text{if } a = b. \\ 0, & \text{otherwise.} \end{cases} \quad (2.4)$$

The EM process would ideally be iterated until convergence, though in practice it is often iterated for some fixed number of times (often 5) to save processing time.

For Portuguese sentence \mathbf{p} and English sentence \mathbf{e} , a set \mathbf{a} of alignments is then defined to have the following probability:

$$p(\mathbf{a}|\mathbf{p}, \mathbf{e}) = \prod_{i=0}^{l_e} t(e_j|p_{a(j)}) \quad (2.5)$$

where $a(j)$ is the index of the word in \mathbf{p} to which e_j is aligned.

Because each word is aligned independently, the probability in Equation 2.5 can be maximized by, for each Portuguese word p , aligning it with

$$\arg \max_{e \in \mathbf{e}} t(e|p)$$

IBM Model 2

IBM Model 2 aligns words based on both translation probability and alignment probability. Thus, for Portuguese sentence \mathbf{p} and English sentence \mathbf{e} , a set \mathbf{a} of alignments is defined to have the following probability:

$$p(\mathbf{a}|\mathbf{p}, \mathbf{e}) = \prod_{j=0}^{l_e} t(e_j|p_{a(j)})a(a(j)|j, l_e, l_p) \quad (2.6)$$

where $a(j)$ is the index of the word in \mathbf{p} to which e_j is aligned, l_e is the length of the English sentence, and l_p is the length of the Portuguese sentence.

Rather than initializing the translation probabilities uniformly (as was done for IBM Model 1), IBM Model 2 builds on IBM Model 1 by using translation probabilities trained from IBM Model 1 as its initial translation probabilities. However, the alignment probabilities (which are the aspect newly added to create Model 2) are initialized uniformly. As in Model 1, an EM process is then used to tune the values of both the alignment probabilities and the translation probabilities.

Again, as with Model 1, each translation probability and alignment probability can be calculated independently of how the other words in the sentences were aligned; thus, each Portuguese word p can simply be aligned with the English word that maximizes p 's component of the product in Equation 2.6.

IBM Model 3

IBM Model 3 adds the notion of fertility on top of IBM Model 2; thus, IBM Model 3 is conditioned on translation probability, alignment probability, and fertility. Under this model, for Portuguese sentence \mathbf{p} and English sentence \mathbf{e} , a set \mathbf{a} of alignments is defined to have the following probability:

$$p(\mathbf{a}|\mathbf{p}, \mathbf{e}) = \prod_{i=0}^{l_p} \phi_i! n(\phi_i|p_i) \prod_{j=0}^{l_e} t(e_j|p_{a(j)})a(a(j)|j, l_e, l_p) \quad (2.7)$$

where $a(j)$ is the index of the word in \mathbf{p} to which e_j is aligned, l_e is the length of the English sentence, l_p is the length of the Portuguese sentence, and ϕ_i is the fertility of Portuguese word p_i —that is, the number of English words (potentially zero) to which p_i is aligned. The $\phi_i!$ term arises from the fact that the fertility component of IBM Model 3 can be considered as first mapping the Portuguese word p to ϕ instances of itself and then aligning the result with the English sentence. This means that, for a given final alignment, there will be $\phi!$ different ways to create the final alignment from this intermediate representation (since there are ϕ instances of p that may be placed in any order).

For a given Portuguese word p , its fertility ϕ is modeled with a distribution $n(\phi|p)$. Model 3 is initialized with the translation and alignment probabilities outputted by Model 2, while the fertility distributions are initialized uniformly; all three distributions are then trained using EM.

IBM Model 4

IBM Model 4 adds the notion of relative alignment, here referred to as distortion. It is therefore conditioned on translation probability, alignment probability, fertility, and distortion. Under this model, for Portuguese sentence \mathbf{p} and English sentence \mathbf{e} , a set \mathbf{a} of alignments is defined to have the following probability:

$$p(\mathbf{a}|\mathbf{p}, \mathbf{e}) = \prod_{i=0}^{l_p} \phi_i! n(\phi_i|p_i) d(a'(i) - a'(i-1)) \prod_{j=0}^{l_e} t(e_j|p_{a(j)}) a(a(j)|j, l_e, l_p) \quad (2.8)$$

where $a(j)$ is the index of the word in \mathbf{p} to which e_j is aligned, $a'(i)$ is the index of the word in \mathbf{e} to which p_i is aligned, l_e is the length of the English sentence, and l_p is the length of the Portuguese sentence.

Model 4 is initialized with the translation probability, alignment probability, and fertility distributions outputted by Model 3, while the distortion probabilities are initialized uniformly. Again, all parameters are then trained using EM. Note that, in the implementation I will be using (namely mgiza (Gao and Vogel 2008)) the distortion probabilities are conditioned over word classes, so that rather than having the term $d(a'(i) - a'(i-1))$ we would have $d(a'(i) - a'(i-1)|c)$ where c is a word class. Ideally, the distortion probabilities would be fully lexicalized (that is, with the probability conditioned over individual words), but data is typically too sparse to do this. Instead, then, words are split into about 100 classes based on distribution using the mkcls tool in the Moses software package; this tool creates word classes by clustering words such that the probability of a string of words based on a class bigram model is as close as possible to the probability of the string of words based on a word bigram model, so that the distortions may be conditioned over these classes (Och 1999). The notion of using word classes is based on the fact that different types of words have varying likelihoods of changing their positions relative to surrounding words when translating from one language to another. For example, there might be a pair of languages that are both SOV but where one language has adjectives appearing before the nouns and the other has adjectives after the noun. In this case, the class-based distortion probabilities would reflect that adjectives are likely to be reordered with respect to the nouns they're near whereas verbs are not likely to be reordered with respect to the nouns they are near.

IBM Model 5

Equation 2.8 does not create a true probability distribution because it allows multiple words from the intermediate representation created after the fertilities have been determined (discussed under Model 3) to be aligned to the same English word, when in fact this should not be allowed; it thus allocates some probability mass to impossible cases. This problem is called *deficiency*, and the main function of IBM Model 5 is to fix deficiency. However, I do not consider Model 5 in this paper

because it is not implemented in mgiza, the alignment toolkit that I use as a basis for many of the probabilities that I use (Gao and Vogel 2008).

IBM summary

Each IBM model introduces some new linguistic aspect of the task of translation. IBM Model 1 basically corresponds to vocabulary matching or word-for-word translation, something like the act of dictionary lookup. IBM Models 2 and 4 bring some syntax into the mix by factoring in sentence-wide word position and local reorderings, respectively. IBM Model 3 allows for the fact that there is not always a one-to-one mapping between words in different languages. (IBM Model 5 does not really lend any new linguistic components, however).

Once training has been completed for the various IBM models, the resulting probability distributions can be combined with a language model for the target language to be used for word-based machine translation. This is the approach used by the popular machine translation toolkit Moses (Koehn et al. 2007).

2.1.2 HMM word alignment

A hidden Markov model (HMM) is a model in which there is some sequence of hidden states paired with a sequence of observations of some non-hidden property. The model is used to generate new observations based on the sequence of hidden states that have already been generated. In theory, all previous hidden states would be considered when choosing the next hidden state, but in practice this approach leads to sparse data problems; thus, typically some order n is chosen such that only the n previous hidden states are considered, and an HMM with such an assumption is called an n^{th} -order HMM. Vogel et al. (1996) propose an HMM-based word alignment model that, like IBM Model 4, focuses on the relative positions of words rather than their absolute positions (as in IBM Model 2). In this model, the source words are considered in sequence, with each one being aligned with some word in the target sentence. This alignment is conditioned on the position to which the previous source word was aligned (as well as the translation probability of the alignment under consideration); thus, the sequence of alignment indices in the target sentence can be viewed as the sequence of hidden states, and this model is therefore a first-order HMM. Thus, for Portuguese sentence \mathbf{p} and English sentence \mathbf{e} , a set \mathbf{a} of alignments is defined to have the following probability:

$$p(\mathbf{a}|\mathbf{p}, \mathbf{e}) = \prod_{j=0}^{l_e} t(e_j|p_{a(j)})d(a(i)|i, l_e, l_p, a(i-1)) \quad (2.9)$$

where $a(i)$ is the index of the word in \mathbf{e} to which p_i is aligned, l_e is the length of the English sentence, and l_p is the length of the Portuguese sentence.

Thus, this model is basically equivalent to IBM Model 2, except for two differences: (i) IBM Model 2 uses absolute positions to condition its alignment probabilities, while the HMM uses relative positions; and (ii) IBM Model 2 does not consider previous alignments (which could be described by calling IBM Model 2 a zeroth-order HMM) while the HMM considers one previous alignment

(meaning that it is a first-order HMM). The HMM generally performs better than IBM Model 2, so it is conventionally used in place of Model 2 in the implementation of word alignment algorithms.

2.1.3 Phrase-based machine translation

When word alignment (whether conducted using the IBM Models, an HMM, or some other method) is used as the basis of a machine translation system, the overall machine translation system is referred to as word-based machine translation because it focuses on taking each word in the source sentence and figuring out what to do with it in the target sentence (whether to delete it, what to translate it as, where to move it, etc.). However, there are newer techniques called phrase-based machine translation, described in Koehn et al. (2003), Och and Ney (2004), and others, which considers the translation of entire phrases rather than individual words, where a phrase is given the loose definition of any sequence of words, meaning that, for example, *ran into the* is a phrase for the purposes of phrase-based machine translation even though no syntactician would call it a phrase. Phrase-based methods generally outperform word-based methods, and they also eliminate much of the clunkiness of the IBM Models; for example, in a word-based model, the fact that the English word *hello* can be translated into the two-word German phrase *guten Tag* must be expressed in the convoluted way of having a high probability that the fertility of *hello* will be two and by having high alignment probabilities between *hello* and *guten* and between *hello* and *Tag*. However, in a phrase-based system, all of that complexity is collapsed by simply assigning a high probability to the translation of the one-word English phrase *hello* and the two-word German phrase *guten Tag*.

The algorithm developed in this paper is aimed specifically at word alignment, and thus it currently applies mainly to word-based machine translation. However, the method of using a pivot language may well be extendible into phrase-based techniques as well.

2.1.4 Neural machine translation

In recent years, neural networks have gained great popularity as a way to create a streamlined, end-to-end machine translation system (e.g. Collobert and Weston. (2008), Cho et al. (2014), Bahdanau et al. (2015)). Neural machine translation often makes use of encoder-decoder networks, which consist of an encoder component that encodes a sequence of inputs (in this case, a sequence of words or even of characters) into an intermediate representation, as well as a decoder component which uses the intermediate representation to generate an output sequence of words (or characters) in the target language. This approach has proven to be quite successful, and it also to some extent realizes the long-dreamt-of interlingua approach to translation (with which there is some intermediate language called an interlingua to which any other language can be translated and from which any other language can be generated) since the encoded input can in theory be fed to a decoder that generates text in any output language. However, neural machine translation is not very conducive to the pivot language techniques discussed here because the pivot language techniques require you to open the hood of the translation program and change some parameters, and neural networks are not conducive to such tinkering. Finding a way to integrate these pivot-based methods with neural techniques may be a fruitful direction for future research given the typically high performance exhibited by neural networks.

2.2 The problem: alignment without training data

All of the previously-discussed approaches to word alignment and machine translation depend crucially on having large quantities of parallel bilingual text in the two languages of interest. Even for the languages for which much data is available, obtaining parallel bilingual text is not always easy because most linguistic data available is monolingual; thus it is only for the most high-resource languages that enough data is available to create a high-performing machine translation. For the several thousand languages that do not meet this criterion, it is very difficult to achieve good translation quality. For many languages, there is barely any data on which to build even a rudimentary translation system. Thus, the question that I address in this paper is how to deal with data shortages in such scenarios.

Chapter 3

Proposal: Pivot-based word alignment

3.1 Outline

My proposal for tackling the task of word alignment when one of the languages is low resource but the other is high-resource is outlined in Figure 3.1. The idea is to first choose a pivot language that will ideally fulfill two criteria: (i) It should be closely related to the low-resource language and (ii) it should be high-resource, in the sense that a reasonably large bilingual corpus exists containing text in the pivot language and in the high-resource target language. By having it be related to the low-resource language, we can use it to infer information about the low-resource language; and by having it be high-resource, we ensure that we have enough information about it to overcome the data shortage of the low resource language.

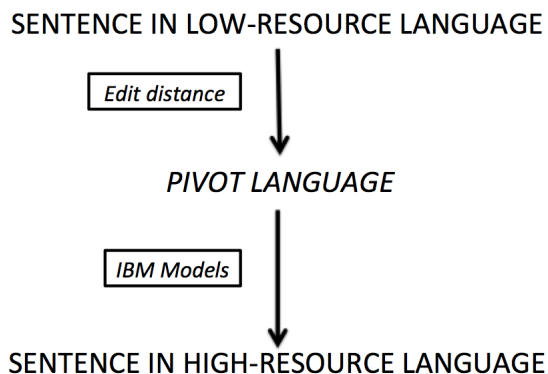


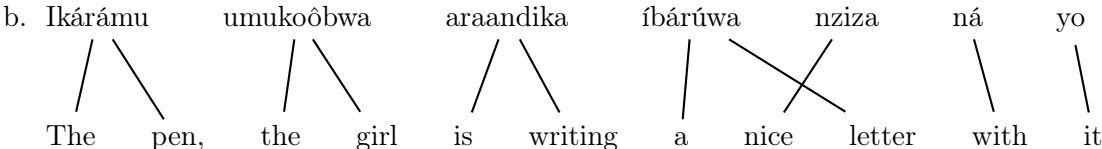
Figure 3.1: Outline of the proposal.

Now that we have chosen the pivot language, we need some way to move from the low-resource language to the pivot language and also some way to move from the pivot language to the high-resource language. In order to move from the low-resource language to the pivot language, I will use edit distance, which quantifies the similarity between strings. By using this technique, I can find which words in the pivot language are likely cognates of the words in our low-resource sentence, which allows for a rough translation into the pivot language. To move from the pivot language to the high-resource language, I will be using probabilities derived from the IBM Models, as with standard

IBM word alignment, because the fact that both the pivot language and the high resource language are high resource means that there will be enough data to calculate these probabilities. There are thus three major components of my model—the pivot language, the edit distance algorithm, and the implementation of the IBM models—and in the experiments section I will test various options for each of these. First, however, I will go through a brief, general example of the algorithm to make its usage clearer.

3.2 Example

Consider the following example, reproduced from (1) and originally from Gerdts and Whaley (1991):

- (2) a. Ikárámu umukoôbwa a-ra-andik-a íbárúwa nziza ná yo.
 pen girl she-PRES-write-ASP letter good with it
 ‘The pen, the girl is writing a nice letter with it.’
- b. Ikárámu umukoôbwa araandika íbárúwa nziza ná yo

 The pen, the girl is writing a nice letter with it

(2a) shows a sentence in Kinyarwanda as well as its translation, while (2b) shows the word alignment between the Kinyarwanda sentence and the English sentence. Suppose that you had been given (2a) and wished to generate the alignment in (2b). This would be done as follows:

- First, identify a pivot language to act as an interface between Kinyarwanda and English. In this case, I choose Swahili, as it is related to Kinyarwanda (they are both Bantu languages) and it is much higher-resource than Kinyarwanda, since it is a language of international commerce.
- Now, consider each Kinyarwanda word in turn. Thus, we would first consider the word *ikárámu* “pen.” Go through all words in the Swahili vocabulary to find the Swahili words that have the smallest edit distance from *ikárámu*. Suppose that the two closest Swahili words found are *karamu* “feast” and *kalamu* “pencil, pen.”
- Now consider each pairing of one of those Swahili words (*karamu* and *kalamu*) with one of the English words in the English sentence and find which pair is most likely to be translations of each other (using the translation probabilities calculated by training the IBM models on some bilingual Swahili/English text). This procedure would find that (*kalamu*, *pen*) is a very likely pairing, while no other pairs are very likely.
- Since *pen* was the English word that yielded the most likely match for our Kinyarwanda word’s Swahili proxy, we now draw the alignment between *ikárámu* and *pen* and then proceed through the rest of the sentence until all of the Kinyarwanda words have been aligned.

This will be the basic framework used for all of the experiments in the following chapter, but many variations on the exact parameters will be tried.

Chapter 4

Experiments

4.1 The languages

For all experiments, the specific task I will tackle is alignment between Spanish and English. Although Spanish is an extremely high-resource language in real life, for these experiments I will be simulating low-resourcedness by not providing the computer with any Spanish training data; thus, the only Spanish that the computer will be exposed to is test data. I chose this path rather than using a truly low-resource language because it is much easier to create a gold standard alignment set (from which one can evaluate experimental success) for a high-resource language. As in real life, English will be treated as a high-resource language in these experiments. As for pivot languages, seven different languages will be tested as potential high-resource pivot languages: Danish, Finnish, French, German, Italian, Portuguese, and Spanish.

4.2 The corpus

Conducting an alignment task requires access to a parallel corpus—that is, the same text written in multiple languages. One popular source of parallel corpora is the proceedings of governmental organizations that conduct their business in multiple languages. For example, much early work in machine translation (e.g., Gale and Church (1993), Chen (1993)) made use of the Canadian Hansards corpus, which contains the Canadian parliament’s proceedings in both French and English. Such a government-derived corpus is attractive for several reasons: It is freely available; its two aligned components are likely to correspond closely in meaning since they were both generated from the same utterance by a parliament member; and it contains a relatively large quantity of data.

Both for training data and for test data, all experiments used the Europarl parallel corpus, another government-based corpus that was created by Koehn (2005). This corpus comes from the proceedings of the European Parliament, a governing body of the European Union. Because of the multinational nature of this body, its proceedings are available in the languages of many EU member states, and thus these proceedings are useful for a project that explores alignments between more than one pair of languages.

Several of the languages involved in this study use characters beyond the 26 letters used in

Portuguese	English
comunico à assembleia que , no final do debate , recebi duas propostas de resolução .	i inform the house that at the end of the debate i have received two motions for resolutions .
por razões de ordem técnica , estas propostas de resolução não poderão , de modo nenhum , ser votadas durante a presente sessão .	for technical reasons , these motions for resolutions can not in any case be put to the vote during this round of voting .
segue-se o período dedicado às intervenções dos deputados que pedem a palavra .	we come to the time for speeches by those members who ask to speak .
como sabem , essas intervenções estão limitadas a um minuto .	as you know , they are limited to one minute each .
senhor presidente , gostaria de o informar que a empresa produtora de enchidos igloomeat sa sokołów polska pediu para ser retirada do anexo xii do tratado de adesão .	mr president , i should like to inform you that the salami producer igloomeat sa sokołów polska has asked to be deleted from appendix 12 to the treaty of accession .

Table 4.1: Example sentence pairs from the Portuguese/English training set.

English. In the corpus, accented Roman characters are encoded as the plain Roman character plus a separate character indicating the accent. This means that, for example, the deletion of an accented vowel would be counted as the deletion of two characters when computing Levenshtein distance.

4.2.1 Training sets

The Europarl corpus is available in the form of twenty different language pairs, where each pair consists of English plus some other European language. Different experiments that I ran made use of different language pairs; the language pairs that were used for training were Spanish/English, Portuguese/English, French/English, Italian/English, German/English, Danish/English, and Finnish/English, while the remaining 13 languages in the corpus were not used in this project. Each training set was lowercased and tokenized before training, and each training set consists of 1 million parallel sentences. Table 4.1 shows a portion of the Portuguese/English training data.

4.2.2 Test set

The test set remained constant across all experiments and consisted of 1,000 lowercased and tokenized parallel sentence pairs from the Spanish/English component of the Europarl corpus. (Note that all training corpora were filtered so as not to include any sentences that are in the test set). The test set also contains a gold standard set of alignments from the NAACL 2006 shared task on statistical machine translation, available at <http://www.statmt.org/wmt06/shared-task/>. These gold standard alignments were generated using automatic methods (the exact methods are not stated), so they can be expected to contain some errors, which makes this test set not ideal for assessing performance on this task of word alignment. However, for my purposes, this training data

Spanish	English	Alignments
considero muy importante que nos hagamos conscientes de que la movilidad social y la integraci3n de la poblaci3n europea debe ir acompa1ada de una armonizaci3n del derecho .	i believe it is very important that we remind ourselves that social mobility and the integration of the people of europe must be accompanied by harmonisation of the law .	0-0 0-1 0-2 1-4 2-5 3-6 4-7 5-8 5-9 6-8 8-10 9-12 10-12 11-11 12-13 13-14 14-15 15-16 16-17 17-18 18-20 19-21 20-22 21-23 21-24 22-24 23-25 24-25 25-26 25-27 26-28 27-29
por ello debe existir una buena colaboraci3n entre las autoridades responsables de la pol3tica presupuestaria y del banco central europeo , que al fin y al cabo es quien determina los tipos de inter3s .	therefore , good cooperation must be brought about between the governments who are responsible for budget policy and the european central bank , which ultimately determines interest rates	0-0 1-0 2-4 3-5 3-6 4-7 5-2 6-3 7-8 8-9 9-10 10-11 10-12 10-13 11-14 13-16 14-15 15-17 16-18 17-21 18-20 19-19 20-22 21-23 22-24 23-24 24-24 25-24 26-24 27-12 28-11 29-25 30-27 31-27 32-27 33-26 34-28
apruebo las orientaciones que se nos proponen .	i am in favour of the proposed guidelines .	0-0 0-1 0-2 0-3 0-4 1-5 2-7 4-6 5-6 6-6 7-8

Table 4.2: Example sentence pairs and their alignments from the English/Spanish test set.

is sufficient because the small levels of inaccuracy associated with the automatic errors in preparing this test set are not significant in the context of a low-resource NLP challenge.

Table 4.2 shows 3 of the test Spanish/English sentence pairs and their gold-standard alignments. Each alignment is a list of pairs of numbers, where the first element of the pair is an index in the Portuguese sentence while the second element is an index in the English sentence. Having two indices occur together in one of these pairs means that the words at those indices are aligned. For example, in the bottom row of the table, the Spanish word *apruebo* (at index 0) is aligned with each of the first 5 English words (composing the string of words *i am in favor of* in indices 0 through 4).

4.3 Measuring success

In NLP tasks, it often is not very clear how to quantify success; for example, in the task of machine translation, the fact that each sentence usually has many possible translations into another language makes it difficult to assess how correct a given translation is. Luckily, however, word alignment is one of the NLP tasks that does have very easy-to-quantify results. I will use three quantitative measures of the performance of each program (namely, precision, recall, and alignment error rate), which are described below.

4.3.1 Precision

Precision is the proportion of alignments predicted by the program that are present in the gold standard. It is calculated with the following formula:

$$\text{precision} = \frac{\text{number of predicted alignments that are correct}}{\text{number of predicted alignments}} \quad (4.1)$$

Precision falls within the range of 0 to 1, where it is best to be as close to 1 as possible.

4.3.2 Recall

Recall is the proportion of alignments present in the gold standard that are predicted by the program. It is calculated with the following formula:

$$\text{recall} = \frac{\text{number of predicted alignments that are correct}}{\text{number of correct alignments}} \quad (4.2)$$

Recall falls within the range of 0 to 1, where it is best to be as close to 1 as possible.

4.3.3 Alignment error rate

Precision and recall are somewhat in opposition: It is easy to achieve a high precision simply by not predicting very many alignments, while it is easy to achieve a high recall by predicting far too many alignments. To reconcile these competing behaviors, a third metric is used which balances precision and recall. It is called alignment error rate (AER), and it is calculated with the following formula:

$$\text{AER} = 1 - \frac{2(\text{number of predicted alignments that are correct})}{\text{number of correct alignments} + \text{number of predicted alignments}} \quad (4.3)$$

AER falls within the range of 0 to 1, where it is best to be as close to 0 as possible.

4.4 Baselines

The following tests are meant to give some basis for understanding what the performance numbers mean for the main tests.

4.4.1 Lower-bound baselines

These baselines are two of the simplest possible alignment algorithms that could be used. Thus, any word alignment program worth its salt should definitely outperform these two (luckily, that's not a very high bar).

Random alignment

Random alignment consists of, for each word in the input, aligning it at random with one word in the output.

Diagonal alignment

When aligning a Spanish sentence with an English sentence, let l be the length of the shorter of the two sentences. Diagonal alignment is defined as aligning the i^{th} English word with the i^{th} Spanish word for every $i \leq l$. (If the sentences are of unequal length, this means that some words at the end of the longer sentence will be unaligned; this procedure could be implemented differently such that those excess words are, for example, all aligned with the last word in the other sentence). It is called diagonal alignment because, in the charts commonly used to depict word alignment (e.g. in Figure 4.1) such an alignment appears as a diagonal line from the upper left hand corner to the lower right hand corner.

4.4.2 Upper-bound baselines

The following alignment methods aim to estimate the upper bound of how well a program could perform on the data.

Using Spanish as a pivot language

We would expect the highest-performing pivot language to be Spanish itself, so we can use Spanish as a pivot language to estimate how well the ideal pivot language would perform. (See Section 4.7 for more details on the experiments dealing with different pivot languages).

Alignment with fast-align

Finally, I used the state-of-the-art word aligner program fast-align, developed as part of the cdec package (Dyer et al. 2010), to train an alignment program on the entire 1 million sentence Spanish-English training corpus, to generate what can be thought of as the best possible word alignments that could be achieved from the given datasets (since this program is state-of-the-art).

4.4.3 Baseline results

The results of these baselines are summarized in Table 4.3. The most important statistic is the alignment error rate, and from these results we can see that a reasonable range for the performance of a word alignment system on this data would be somewhere in the range from 0.360 to 0.819, since even a very naive method can achieve 0.819 while even the best-case method does not do better than 0.360. The fact that the diagonal alignment performs so much better than random is mainly due to the fact that English and Spanish have broadly similar word orders (for example, both are subject-initial), so a lot of low-hanging fruit can be grabbed by assuming that the English and Spanish words are in the same order. Thus, if our language pair differed more in high-level syntax (for example, if one language were OVS and the other VSO), then the diagonal alignment would likely work significantly less well. The fast-align performance of 0.288 can be considered a rough upper bound for performance. 0 is the theoretically best AER that can be achieved, but there are a few reasons why the best-case scenario is so far from 0. First, though this data set is moderately large, it is still not large enough to reach the truly state-of-the-art performance of systems trained

Method	Precision	Recall	AER
Random	0.056	0.049	0.948
Diagonal	0.203	0.163	0.819
Spanish as pivot	0.682	0.602	0.360
fast-align	0.744	0.683	0.288

Table 4.3: Results of baseline alignments.

on more data. Second, and perhaps more importantly, the test set is not actually a gold standard; its alignments have been generated by automatic methods, not by humans, so these alignments may well be wrong in places. Indeed, the task of word-alignment is not always clear cut, so even in human-generated data there would be some level of indeterminacy. This is why, even though the theoretical range of AER values is 0 to 1, a more practical range to keep in mind when assessing the results that follow would be 0.360 to 0.819.

4.5 Edit distance experiments

I tested several different methods for measuring edit distance. Sections 4.5.1 through 4.5.4 describe these different methods, and Section 4.5.5 discusses the results.

4.5.1 Levenshtein edit distance

The first edit distance metric that I used was the standard Levenshtein edit distance (Levenshtein 1966; Wagner and Fischer 1974). Levenshtein edit distance focuses on three possible operations that can be performed on a string of characters:

1. **Insertion:** The insertion of a new character into the string.
2. **Deletion:** The deletion of a character already present in the string.
3. **Substitution:** The substitution of some new character for a character already in the string.

The Levenshtein edit distance between two words w_1 and w_2 is then defined as the total number of insertions and/or deletions and/or substitutions that must be made to transform w_1 into w_2 . Table 4.4 contains some examples of word pairs and the Levenshtein edit distance ($dist_L(w_1, w_2)$) between them. In addition, to show how this algorithm works in the context of the actual data, Table 4.5 shows a few Spanish words in the test set along with the four Portuguese words from the Portuguese training set that have the smallest edit distance from the Spanish word.

Note that there are many possible ways to transform one string into another (in fact, there are infinitely many ways). For example, *emanate* could be turned into *manatee* by substituting an *m* for the first *e*, then substituting an *a* for the *m*, then substituting an *n* for the *a*, then substituting an *a* for the *n*, then substituting a *t* for the *a*, then substituting an *e* for the *t*—making a total of 6 operations. However, the same transformation could instead be effected by deleting the initial *e* and adding an *e* at the end, which is only two operations. Therefore, note that whenever I refer

w_1	w_2	$dist_L(w_1, w_2)$	Operations performed
stephen king	stephen hawking	3	insert(h), insert(a), insert(w)
lemony snicket	jiminy cricket	5	sub(l, j), sub(e, i), sub(o, i), sub(s, c), sub(n, r)
lewis black	louis ck	5	sub(e, o), sub(w, u), del(b), del(l), del(a)
jim carrey	john kerry	6	sub(i,o), insert(h), sub(m, n), sub(c, k), sub(a, e), del(e)

Table 4.4: Examples of Levenshtein edit distance.

Spanish word	Neighbor 1	Neighbor 2	Neighbor 3	Neighbor 4
ciudades ‘cities’	ciudades (1) ‘cities’	ciudad (2) ‘city’	saudades (2) ‘miss’	idades (2) ‘ideas’
seis ‘six’	seis (0) ‘six’	sei (1) ‘know’	deis (1) ‘you are’	leis (1) ‘laws’
ocasiona ‘causes’	ocasiona (0) ‘causes’	ocasiona (1) ‘occasion’	ocasional (1) ‘casual’	ocasionam (1) ‘occasion’
otro ‘other’	ouro (1) ‘gold’	outro (2) ‘other’	otros (1) ‘other’	oro (1) ‘gold’
arriesgarnos ‘to risk’	arriscarnos (3) ‘to risk’	arriscar-nos (3) ‘to argue’	arrastarnos (4) ‘drag’	apressarmo (4) ‘hurry up’

Table 4.5: Examples of Levenshtein neighbors (distances from the Spanish words are in parentheses)

to the edit distance between two strings in this paper, it is intended to refer to the *minimum* edit distance between them—so in this case $dist_L(emanate, manatee) = 2$.

For this edit distance experiment, as with all edit distance experiments in this section, the pivot language is Portuguese and the model for alignment probability is simply the IBM Model 1 translation probabilities. In addition, in order to compare edit distance algorithms under stricter conditions, I am stipulating that the only Portuguese word to consider is the first choice in terms of the closest Levenshtein neighbor to the Spanish word under consideration.

4.5.2 Feature-based edit distance

This section details two approaches to edit distance in which the basic aim is to alter the penalties associated with the Levenshtein operations of insertion, deletion, and substitution based on the phonological properties of the characters involved.

The assumption underlying this method is that, when two cognates differ in some of the phonemes they contain, they are likely to differ in phonologically sensible ways. For example, it is more likely that one cognate will contain a *d* where its partner contains a *t* than it is for one cognate to contain a *d* where the other contains a *u*. If this assumption is true, an edit distance algorithm that encodes some phonological information may be more successful at identifying cognates than the basic Levenshtein algorithm.

Vowel/consonant approach

In this model, the penalty for substituting a vowel for a consonant, or for substituting a consonant for a vowel, is greater than that for substituting a vowel for a vowel or for substituting a consonant for a consonant. Specifically, this model works almost exactly like the basic Levenshtein—with a penalty of 1 for an insertion or a deletion or for a substitution that does not change a vowel to a consonant or vice versa—but the penalty for substituting a vowel for a consonant (or vice versa) is modified to be greater than one. Table 4.6 shows some examples of the closest Portuguese neighbors of a few words in the Spanish test set where the vowel/consonant method is used, which a vowel/consonant substitution penalty of 2 and a penalty of 1 for all other operations.

More features

In this model, the penalty for any operation is (in theory) set to be equal to the number of phonological features that change when that operation occurs. For example, substituting a *d* for a *t* incurs a penalty of 1 because a single feature (namely, voicing) has changed, while substituting a *b* for a *t* incurs a penalty of 2 because a two features (namely, voicing and place) have changed. In practice, it is impossible to enact this approach rigorously because of the fact that orthography does not map cleanly to phonology and does not have consistent phonological properties across languages; that is, it is impossible to definitively state what phonological features a given character possesses, and because of that it is impossible to determine exactly how many features change in a given operation. Therefore, many of the weights that ended up being used for this method were somewhat hand-wavy. Table 4.7 shows the weights used for all substitutions; meanwhile, each insertion or deletion

Spanish word	Neighbor 1	Neighbor 2	Neighbor 3	Neighbor 4
ciudades ‘cities’	ciudade (2) ‘capital’	ciudades (2) ‘cities’	saudades (2) ‘miss’	idades (3) ‘ideas’
seis ‘six’	seis (0) ‘six’	sais (1) ‘salts’	deis (1) ‘you are’	reis (1) ‘kings’
ocasiona ‘causes’	ocasiona (0) ‘causes’	ocasiona (1) ‘occasion’	ocasional (1) ‘casual’	ocasionam (1) ‘occasion’
otro ‘other’	otto (1) ‘otto’	oro (1) ‘gold’	otros (1) ‘other’	coro (2) ‘choir’
arriesgarnos ‘to risk’	arriscarnos (4) ‘to risk’	arriscar-nos (4) ‘to argue’	arrastarnos (5) ‘drag’	apressarnos (5) ‘hurry’

Table 4.6: Examples of vowel/consonant neighbors (distances from the Spanish words are in parentheses) .

was given a weight of 3 (under the assumption that each character has roughly three major features, and thus inserting or deleting that character amounts to inserting or deleting three features). Table 4.8 shows some examples of the closest neighbors of a few Spanish words when using this method to compute edit distance.

4.5.3 Embedding-based edit distance

Embedding-based edit distance works by creating an embedding for each character (that is, a vector that represents the character). Once such embeddings have been created, characters can be compared by determining the cosine between their respective embedding vectors, which is given by the following equation:

$$\text{cosdist}(c_1, c_2) = \frac{\vec{c}_1 \cdot \vec{c}_2}{\|\vec{c}_1\| \|\vec{c}_2\|} \quad (4.4)$$

where \vec{c}_1 and \vec{c}_2 are the vector embeddings of c_1 and c_2 . The penalty for substituting c_1 for c_2 is defined as $\text{cosdist}(c_1, c_2)^{-1}$ (the negative exponent is there because the cosine is greater for more similar vectors, whereas we want a smaller penalty for more similar vectors), while I tried two different methods for determining the penalty for inserting or deleting character c_1 :

- **Average method:** The penalty for inserting or deleting c_1 is the average of all substitution penalties; this can be expressed as $\frac{\sum_{c_1, c_2} \text{cosdist}(c_1, c_2)}{\|C\|}$, where C is the alphabet.
- **Ngram method:** For this method, in addition to having embeddings for all the characters, an embedding is also created for the empty character ϵ . The penalty for inserting or deleting character c_1 is then $\text{cosdist}(c_1, \epsilon)$.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	4	4	4	1	4	4	4	1	4	4	4	4	4	1	4	4	4	4	4	1	4	4	4	2	4
b	4	0	2	1	4	3	1	3	4	2	2	2	1	2	4	2	2	2	3	2	4	2	2	3	2	2
c	4	2	0	2	4	2	1	2	4	3	1	3	3	3	4	1	3	3	2	1	4	1	3	3	2	2
d	4	1	2	0	4	3	1	3	4	2	2	2	2	1	4	2	2	2	3	2	4	2	2	3	2	2
e	1	4	4	4	0	4	4	4	1	4	4	4	4	4	1	4	4	4	4	4	1	4	4	4	2	4
f	4	3	2	3	4	0	3	1	4	3	2	3	3	3	4	2	3	3	2	2	4	1	3	3	2	3
g	4	1	1	1	4	3	0	3	4	2	1	2	2	2	4	2	3	2	3	2	4	2	3	3	2	2
h	4	3	2	3	4	1	3	0	4	2	1	2	2	2	4	2	3	3	1	2	4	3	3	3	2	2
i	1	4	4	4	1	4	4	4	0	4	4	4	4	4	1	4	4	4	4	4	1	4	4	4	2	4
j	4	2	3	2	4	3	2	2	4	0	3	2	2	2	4	3	3	2	3	3	4	2	3	3	2	2
k	4	2	1	2	4	2	1	1	4	3	0	3	3	3	4	1	2	3	2	1	4	3	3	3	2	3
l	4	2	3	2	4	3	2	2	4	2	3	0	2	1	4	3	3	1	3	2	4	2	1	3	2	1
m	4	1	3	2	4	3	2	2	4	2	3	2	0	1	4	2	3	2	3	3	4	2	1	3	2	2
n	4	2	3	1	4	3	2	2	4	2	3	1	1	0	4	3	3	2	3	2	4	2	2	3	2	1
o	1	4	4	4	1	4	4	4	1	4	4	4	4	4	0	4	4	4	4	4	1	4	4	4	0	4
p	4	2	1	2	4	2	2	2	4	3	1	3	2	3	4	0	3	3	2	1	4	3	3	3	2	3
q	4	2	3	2	4	3	3	3	4	3	2	3	3	3	4	3	0	3	3	3	4	3	3	3	2	3
r	4	2	3	2	4	3	2	3	4	2	3	1	2	2	4	3	3	0	3	3	4	3	2	3	2	2
s	4	3	2	3	4	2	3	1	4	3	2	3	3	3	4	2	3	3	0	1	4	2	3	2	2	1
t	4	2	1	2	4	2	2	2	4	3	1	2	3	2	4	1	3	3	1	0	4	3	3	3	2	2
u	1	4	4	4	1	4	4	4	1	4	4	4	4	4	1	4	4	4	4	4	0	4	4	4	0	4
v	4	2	1	2	4	1	2	3	4	2	3	2	2	2	4	3	3	3	2	3	4	0	1	3	2	3
w	4	2	3	2	4	3	3	3	4	3	3	1	1	2	4	3	3	2	3	3	4	1	0	3	2	3
x	4	3	3	3	4	3	3	3	4	3	3	3	3	3	4	3	3	3	2	3	4	3	3	0	2	2
y	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	2
z	4	2	2	2	4	3	2	2	4	2	3	1	2	1	4	3	3	2	1	2	4	3	3	2	2	0

Table 4.7: Penalties for feature-based edit distance.

Spanish word	Neighbor 1	Neighbor 2	Neighbor 3	Neighbor 4
ciudades ‘cities’	ciudades (3) ‘cities’	vaidades (3) ‘vanities’	saudades (3) ‘miss’	cuidadas (3) ‘cared for’
seis ‘six’	seis (0) ‘six’	sais (1) ‘salts’	seus (1) ‘their’	sois (1) ‘you are’
ocasiona ‘causes’	ocasiona (0) ‘causes’	ocasiona (1) ‘occasion’	ocasional (3) ‘casual’	ocasionam (3) ‘occasion’
otro ‘other’	adro (2) ‘churchyard’	oslo (2) ‘oslo’	otto (3) ‘ottor’	oro (3) ‘gold’
arriesgarnos ‘to risk’	arriscarnos (5) ‘to risk’	arriscar-nos (7) ‘to argue’	arrastarnos (7) ‘drag’	arriscaremos (8) ‘I will risk’

Table 4.8: Examples of feature-based neighbors (distances from the Spanish words are in parentheses) .

For these tests, all characters beyond the 26 Roman letters were collapsed into a single ‘OTHER’ character, whose embedding was calculated as if it were one consistent character. I tested two different approaches to generating embeddings, described below.

These embedding-based edit distances are founded upon two assumptions: First, as with the feature-based edit distance methods in Section 4.5.2, these methods assume that, across a pair of cognates, it is more likely to substitute one character for a character phonologically similar to it than for a character that is not very phonologically similar to it. Secondly, the embedding-based methods make the further assumption that the distribution of a character can give an accurate portrayal of the character’s phonological nature. Distributional facts certainly can shed light on the phonological properties of a speech sound; for example, Peperkamp et al. (2006) created an algorithm that was highly effective at determining which sounds were allophones vs. distinct phonemes based on the distributions of those sounds. Despite this success, however, it is not necessarily the case that distributional evidence extends its utility to the task of cognate determination.

Count-based embeddings

In this method, the creation for the embedding of a character c begins by considering every occurrence of c in the training set and by looking at the other characters that occur near c in these occurrences within a window of a certain size (I tested two different window sizes—a window of size 3, meaning just c and the two characters immediately adjacent to it, and a window of size 5, meaning c as well as two characters on either side). The embedding for c is then a vector with dimensionality equal to the number of characters in the character set and with the dimension corresponding to character c' equal to the number of times that c' occurs within the window of characters near c in the training data. The embedding for the empty character ϵ (which is needed when the ngram method for assigning insertion/deletion penalties is used) is created by imagining that it occurs between every two characters in the training set and creating its distributional vector accordingly.

Ten most similar pairs	Ten least similar pairs	Selected other pairs
(d,m) 0.973	(q,x) 0.082	(m,n) 0.825
(a,o) 0.970	(e,x) 0.098	(d,t) 0.779
(a,e) 0.967	(o,x) 0.128	(s, t) 0.714
(k,w) 0.965	(a,x) 0.141	(n,t) 0.745
(e,o) 0.964	(i,q) 0.211	(f,v) 0.888
(l,r) 0.957	(h,q) 0.225	(m,g) 0.884
(m,r) 0.954	(q,v) 0.227	(o,t) 0.475
(d,w) 0.951	(q,u) 0.254	(a,n) 0.462
(c,f) 0.945	(e,z) 0.254	(s,g) 0.732
(d,f) 0.945	(t,q) 0.278	(b,l) 0.851

Table 4.9: Selected substitution edit distances as determined using a window of 3 with the count-based method.

Table 4.9 shows some of the cosine similarities between characters that were derived using the resulting embeddings, and Table 4.10 shows some examples of the closest Portuguese neighbors to selected Spanish words under this formalism. The weights illustrated in Table 4.9 do seem to make at least some phonological sense, but the weights certainly are messy and are far from a perfect representation of phonological relatedness. For example, *m* and *g* are deemed more similar under this system than *d* and *t*, and *s* is judged to be more similar to *g* than to *t*.

char2vec-based embeddings

Inspired by the great success of the word2vec algorithm from Mikolov et al. (2013) at creating semantically sensible embeddings for words, I apply the word2vec algorithm to characters in an attempt to create phonologically sensible embeddings for words. By analogy with the name *word2vec*, I refer to this technique as *char2vec*. Like the count-based embedding method described above, the char2vec algorithm begins by considering a window of a fixed size around every instance of character *c*; again, I tested windows of size 3 and 5. For word2vec, larger windows are typically used, but given how many fewer characters there are than there are words, a smaller window size seemed sensible for the char2vec experiments because, unlike with word2vec, the cooccurrence vectors for char2vec are not at all sparse and thus there is little need to look farther away from the target character to further populate the cooccurrence vector. It could potentially be desirable to use a larger window to capture long-distance phonological dependencies such as vowel harmony, but I do not attempt this here.

Once the desired windows around different occurrences of *c* have been established, then a neural network is used to generate the vector embedding for *c*. The neural network used is a simple, feed-forward network with an input layer and an output layer both having dimensionality equal to the number of characters in the character set, and with a single hidden layer of dimensionality 16. The network is then trained using either the continuous bag of words (CBOW) method¹ or the skip-

¹I am not quite zealous enough to start calling this the CBOC (continuous bag of characters) method.

Spanish word	Neighbor 1	Neighbor 2	Neighbor 3	Neighbor 4
ciudades 'cities'	miudezas (8.36)	firmadas (8.58)	causadas (8.58)	firmados (8.58)
	'offal'	'signed'	'caused'	'signed'
seis 'six'	seis (4)	sais (4.03)	sois (4.04)	deis (4.07)
	'six'	'salts'	'you are'	'you are'
ocasiona 'causes'	ocasiona (8)	ocasiona (8.03)	emociona (8.30)	avaliara (8.57)
	'causes'	'occasion'	'excited'	'evaluate'
otro 'other'	etna (4.17)	evro (4.23)	otto (4.25)	erro (4.29)
	'etna'	'will'	'etna'	'error'
arriesgarnos 'to risk'	aprendermos (13.26)	enfrentarnos (13.76)	estipularmos (13.78)	omnipotentes (8.37)
	'grasped'	'face'	'stipulate'	'omnipotent'

Table 4.10: Examples of count-based neighbors (distances from the Spanish words are in parentheses) .

gram method. In both methods, the network is trained on input/output pairs where both the input and the output consist of one-hot vectors that each represent a single character. In the CBOW method, the input one-hot vector represents one of the characters within the window of the target character while the output one-hot vector represents the target character, whereas the skip-gram method swaps which character is the input and which is the output. Once the network finishes training, the trained weight matrix used to transition from the input layer to the hidden layer is used to generate the embeddings for all of the characters. Specifically, for the character at index i in the input vector, its embedding will be row i of the weight matrix.

Table 4.11 shows some of the cosine similarities between characters that were derived using the resulting embeddings, and Table 4.12 shows some examples of the closest Portuguese neighbors to selected Spanish words under this formalism. As with the count-based weights in Table 4.9, these weights overall seem roughly sensible, but there are some weights that seem not to have much phonological sense; for example, s and t are not deemed to be very similar under this system.

4.5.4 Cognate-based edit distance

The embedding-based methods in the previous section all derive their embeddings from a single training language (which was Portuguese in the examples given). Here I try a different method that utilizes some cross-linguistic information from three relatives of Spanish, namely Portuguese, Italian, and French (it remains the case that no Spanish data is being used to train). The idea behind this approach is to identify cognates amongst Portuguese, Italian, and French and to use those cognates to determine which phonological differences are likely to be present in cognate pairs

Ten most similar pairs	Ten least similar pairs	Selected other pairs
(l,r) 0.967	(i,q) -0.152	(m,n) 0.681
(d,m) 0.949	(n,q) -0.091	(d,t) 0.644
(m,r) 0.941	(h,q) -0.091	(s, t) 0.396
(l,m) 0.924	(q,u) -0.071	(n,t) 0.563
(a,e) 0.923	(q,v) -0.023	(f,v) 0.874
(e,o) 0.923	(p,x) -0.019	(m,g) 0.850
(v,z) 0.921	(b,q) -0.015	(o,t) 0.339
(l,z) 0.921	(q,x) -0.013	(a,n) 0.455
(r,d) 0.910	(q,w) -0.005	(s,g) 0.537
(l,g) 0.907	(r,q) 0.020	(b,l) 0.797

Table 4.11: Selected substitution edit distances as determined using a window of 3 with the CBOW method.

Spanish word	Neighbor 1	Neighbor 2	Neighbor 3	Neighbor 4
ciudades ‘cities’	causados (4.05) ‘caused’	saudades (4.05) ‘miss’	causadas (4.06) ‘casued’	saudados (4.06) ‘greeted’
seis ‘six’	seis (0) ‘six’	sois (2.01) ‘you are’	sais (2.01) ‘salts’	deis (2.02) ‘you are’
ocasiona ‘causes’	ocasiona (0) ‘causes’	ocasiona (4.01) ‘occasion’	emociona (4.06) ‘excited’	adiciona (4.08) ‘adds’
otro ‘other’	otto (2.02) ‘otto’	evro (2.03) ‘will’	etna (2.03) ‘etna’	erro (2.03) ‘error’
arriesgarnos ‘to risk’	arregalarnos (6.18) ‘opened’	enfrentarnos (6.19) ‘face’	articularnos (6.19) ‘articulate’	aprendernos (6.19) ‘seize’

Table 4.12: Examples of CBOW-based neighbors (distances from the Spanish words are in parentheses) .

and which are not.

The success of this approach depends on the assumption that the types of sound changes that occur between some pairs of languages within a language family are similar to the types of sound changes that occur between other pairs of languages in that language family. This assumption is by no means necessarily true; indeed, each language pair will almost certainly have some systematic sound changes between its members that are not represented in any other language pairs. However, perhaps it is the case that there will be some broader trends that cut across many members of a family.

The following criteria were used to generate training examples for this experiment; positive examples were identified by finding any pairs (w_1, w_2) that satisfied criteria (i), (ii), (iii), and (iva), while negative examples were identified by finding any pairs (w_1, w_2) that satisfied criteria (i), (ii), (iii), and (ivb):

- (i) w_1 and w_2 are from different languages.
- (ii) The Levenshtein edit distance between w_1 and w_2 is greater than 0 but less than some specified amount d .
- (iii) Both w_1 and w_2 are at least four characters long.
- (iv) a. The most likely English translation of w_1 is the same as the most likely English translation of w_2 .
b. The cosine similarity between the GloVe embeddings of the most likely English translation of w_1 and w_2 is less than 0.5.

For criterion (i), the different languages that were considered were Portuguese, French, and Italian, which are all of the Romance languages (besides the test language of Spanish) being considered in this paper. For criterion (ii), I ran the experiment both with $d = 1$ and with $d = 2$. Criterion (iii) is included because a low Levenshtein edit distance does not mean much for very short words—for example, any two two-letter words will have an edit distance of at most 2, but this by no means implies that all two-letter words are cognates of each other. For criterion (iv), the most likely English translation of a word is identified based on the IBM Model 1 translation probabilities generated by running mgiza on the bilingual training sets. Finally, for criterion (ivb), I used the GloVe embeddings from Pennington et al. (2014) as a metric for determining semantic similarity; words with a cosine similarity less than 0.5 tend not to be very semantically similar, so this criterion is intended to ensure that the negative examples are not cognates despite being phonologically similar, while criterion (iva) is intended to find positive examples by identifying words that both appear phonologically similar and have similar meanings.

When d from criterion (ii) was specified to be 1, this criteria generated 8,718 positive examples and 25,440 negative examples, while having $d = 2$ generated 27,744 positive examples and 448,746 negative examples. I restricted the number of negative examples to be equal to the number of positive examples in each case, so that there ended up being both 8,718 positive examples and 8,718 negative examples when $d = 1$ and 27,744 positive examples and 27,744 negative examples when $d = 2$. Table 4.13 shows some of the positive and negative example pairs generated when $d = 1$.

Word 1	Word 2	Word 1	Word 2
afgane (It.) "afghan"	afghane (Fr.) "afghan"	rendo (It.) "realise"	lendo (Port.) "reading"
stupide (Fr.) "stupid"	stupido (It.) "stupid"	colmando (It.) "closing"	comando (Port.) "command"
fortemente (It.) "strongly"	fortement (Fr.) "strongly"	eternamente (It.) "eternally"	externamente (Port.) "externally"
indissociabile (It.) "inseparable"	indissociable (Fr.) "inseparable"	monge (Port.) "monk"	ronge (Fr.) "plaguing"
serviu (Port.) "served"	servi (Fr.) "served"	ombros (Port.) "shoulders"	ombres (Fr.) "shadows"
propague (Port.) "spread"	propage (Fr.) "spread"	paute (Port.) "transparent"	faute (Fr.) "fault"
discriminata (It.) "against"	discriminada (Port.) "against"	marini (It.) "marine"	martini (Fr.) "martini"
pressione (It.) "pressure"	pressionem (Port.) "pressure"	verde (It.) "paper"	verse (Fr.) "pays"
perdere (It.) "lose"	perdre (Fr.) "lose"	mentis (It.) "mindset"	mentir (Fr.) "lie"
finali (It.) "final"	finale (Fr.) "final"	stelle (It.) "stars"	telle (Fr.) "such"

Table 4.13: Some of the examples used for training the cognate-based edit distance. The table on the left shows positive examples (pairs deemed to be cognates), while the table on the right shows negative examples (pairs deemed not to be cognates).

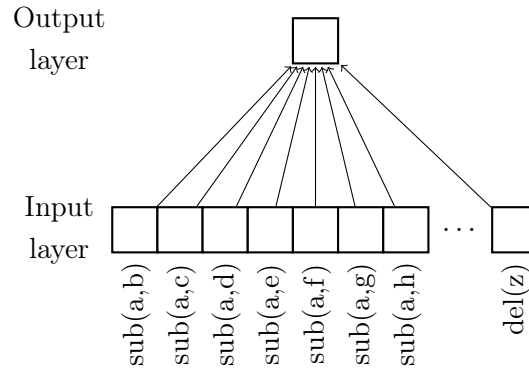


Table 4.14: The architecture of the model used to train the penalties for the cognate-based edit distance.

These examples were then used to train weights for each possible editing operation of insertion, deletion, or substitution. There are 27 characters for which to learn weights (the 26 alphabetic characters plus an OTHER character into which all other characters were collapsed); thus there are $\binom{27}{2}$ possible substitutions that can be made. Because it makes sense for these edit distances to be symmetrical, it was deemed that the penalty for inserting a character should be the same as the penalty for deleting that character, so there were also 27 possible insertion/deletion operations. Thus, there are a total of $\binom{27}{2} + 27 = 378$ operations for which to learn weights. For learning these weights, each training pair (w_1, w_2) was represented as a 378-dimensional vector, where each dimension represented one of the possible string operations. This vector was a 1 at each dimension where the corresponding operation occurred between w_1 and w_2 but 0 otherwise; this means that each vector was 1-hot in the experiment where $d = 1$ or 2-hot in the experiment where $d = 2$ (except that, in a few cases where $d = 2$, both operations that occurred were the same, in which case the vector had one dimension equal to 2 and all the rest equal to 1).

These vectors were then used to train a neural network which consisted only of an input layer and an output layer. The input layer consisted of the 378-dimensional vector representing a given training pair, while the output layer was 1-dimensional and consisted of a 0 if the pair was a positive example or a 1 if it was a negative example. This architecture is illustrated in Table 4.14. The weights learned from training this network were then used as the penalties for each insertion, deletion, or substitution operation when computing cognate-based alignment.

Table 4.15 shows some of the weights that were learned for substitution pairs, and Table 4.16 shows the closest Portuguese neighbors for several words in the Spanish test set, computed using this cognate-based method.

4.5.5 Edit distance results

Results of experiments

Recall from Chapter 3 that there are three basic components that may be varied for my word alignment approach, namely the edit distance algorithm, the pivot language, and the alignment model. For all edit distance experiments, the pivot language was set to be Portuguese and the

Ten most similar pairs	Ten least similar pairs	Selected other pairs
(j,y) -0.480	(q,d) 0.612	(m,n) 0.211
(i,b) -0.435	(r,q) 0.601	(d,t) 0.229
(q,u) -0.426	(p,k) 0.589	(s, t) 0.208
(x,e) -0.338	(b,q) 0.584	(n,t) 0.222
(i,x) -0.322	(p,w) 0.584	(f,v) -0.103
(u,w) -0.294	(w,f) 0.582	(m,g) 0.156
(c,x) -0.288	(b,k) 0.538	(o,t) -0.225
(s,x) -0.254	(j,r) 0.536	(a,n) -0.077
(f,o) -0.253	(t,q) 0.532	(s,g) 0.185
(i,j) -0.248	(f,m) 0.530	(b,l) 0.379

Table 4.15: Selected substitution edit distances as determined using a cognate-based method.

Spanish word	Neighbor 1	Neighbor 2	Neighbor 3	Neighbor 4
ciudades ‘cities’	ciudades (99.9) ‘cities’	ciudad (199.6) ‘city’	cidade (199.8) ‘city’	idades (199.8) ‘ages’
seis ‘six’	seis (0) ‘six’	sis (99.8) ‘sis’	sais (99.8) ‘salts’	seio (99.8) ‘chest’
ocasiona ‘causes’	ocasiona (0) ‘causes’	ocasiona (99.8) ‘occasion’	ocasional (99.9) ‘occasional’	ocasionar (100.0) ‘to cause’
otro ‘other’	otros (99.8) ‘other’	oro (99.9) ‘gold’	outro (99.9) ‘other’	ouro (100.2) ‘gold’
arriesgarnos ‘to risk’	arriesgar-nos (299.6) ‘risk’	arriesgarnos (300.09) ‘to risk’	arrogarnos (399.6) ‘to boast’	carregarnos (399.8) ‘we load’

Table 4.16: Examples of cognate-based neighbors (distances from the Spanish words are in parentheses) .

alignment model was held constant at IBM Model 1, which uses translation probabilities and nothing else to determine alignments. Finally, even after the edit distance algorithm has been chosen, there is one other edit-distance-related variable that can be varied, namely the number of closest edit distance neighbors to consider when making the alignment. For all of the initial experiments, I held this variable at 1. I tested each edit distance algorithm within these constraints, and the results are summarized in Table 4.17

In Table 4.17, none of the more complex methods outperformed basic Levenshtein edit distance. In order to see if this was perhaps because there was too great a range amongst the penalties in the new methods, I reran these tests with various smoothing factors. For smoothing factor s , this simply worked by adding s to every penalty in the edit distance algorithm, which decreases the percentage difference between these penalties. For the vowel/consonant edit distance, I only ran these experiments on the version with $vc = 2$, since that performed better in the preliminary tests. In addition, of the 12 embedding-based methods, I only ran the smoothing tests on the CBOW edit distance with $w = 3$ and the ngram method of determining insertion/deletion weights, as this was the best performing of the twelve. For each of these methods, I re-ran the tests with smoothing factors of 0.1, 1, 5, 10, and 100. For each of these methods, Table 4.18 shows the optimal smoothing factor that was found, along with the performance associated with that smoothing factor.

Once the various methods have had their weights smoothed, the highest-performing edit distance algorithm is the cognate-based edit distance, which does roughly 1% better than basic Levenshtein edit distance.

As a final edit-distance-related test, I varied the number of edit distance neighbors considered when aligning (in all previously discussed experiments, this was held constant at 1). The results of these experiments are in Table 4.19. Once more neighbors were considered, both the CBOW embedding-based edit distance and the vowel/consonant-based edit distance overtook basic Levenshtein edit distance, but cognate-based edit distance remained the highest performing. Also, in general, performance improved as more neighbors were considered, but the improvements in performance quickly leveled off above about 10 neighbors; in fact, the overall best performance observed was cognate-based edit distance with 10 neighbors considered, which outperformed cognate-based edit distance with 100 neighbors considered.

Discussion of edit distance results

In general, the more advanced edit distance algorithms did not bring much improvement over the basic Levenshtein edit distance. There are a few possible reasons for this: First, each new method relies on some assumptions that are not necessarily true. Feature-based edit distance and embedding-based edit distance both assume that phonological similarity is an effective determiner of whether two words are likely cognates, but phonological similarity may not be enough; moreover, both of these methods also assume that orthography can effectively indicate phonological similarity, but orthography and phonology can have a very strained relationship (as popularly illustrated by the English words *cough*, *rough*, *though*, and *through*).

On top of those assumptions, embedding-based edit distance assumes that distributional similarity implies phonological similarity, but in fact there are some reasons to suppose the contrary.

Method	Precision	Recall	AER
Levenshtein edit distance	0.321	0.284	0.698
Vowel/consonant edit distance with a vowel/consonant penalty of 2	0.315	0.279	0.704
Vowel/consonant edit distance with a vowel/consonant penalty of 3	0.305	0.269	0.714
Feature-based edit distance with more features	0.244	0.215	0.771
Count-based edit distance; $w = 5$, method = average	0.218	0.192	0.796
Count-based edit distance; $w = 3$, method = average	0.249	0.219	0.767
Count-based edit distance; $w = 5$, method = ngram	0.213	0.188	0.800
Count-based edit distance; $w = 3$, method = ngram	0.206	0.182	0.807
CBOW edit distance; $w = 5$, method = average	0.247	0.218	0.768
CBOW edit distance; $w = 3$, method = average	0.228	0.202	0.786
CBOW edit distance; $w = 5$, method = ngram	0.220	0.194	0.794
CBOW edit distance; $w = 3$, method = ngram	0.261	0.230	0.755
Skip-gram edit distance; $w = 5$, method = average	0.231	0.204	0.783
Skip-gram edit distance; $w = 3$, method = average	0.223	0.197	0.791
Skip-gram edit distance; $w = 5$, method = ngram	0.234	0.199	0.789
Skip-gram edit distance; $w = 3$, method = ngram	0.201	0.177	0.811
Cognate-based edit distance; $d = 1$	0.057	0.050	0.946
Cognate-based edit distance; $d = 2$	0.045	0.040	0.957

Table 4.17: Preliminary results for various edit distance algorithms.

Method	Optimal smoothing factor	Precision	Recall	AER
Levenshtein	0	0.321	0.284	0.698
VC	100	0.322	0.285	0.698
Feature	100	0.317	0.281	0.702
CBOW	10	0.319	0.282	0.701
Cognate, $d = 1$	10	0.329	0.291	0.691
Cognate, $d = 2$	1	0.332	0.294	0.688

Table 4.18: Smoothed results for various edit distance algorithms.

Method	$n = 1$	$n = 5$	$n = 10$	$n = 100$
Levenshtein	0.698	0.675	0.673	0.672
VC	0.698	0.672	0.670	0.669
Feature	0.702	0.679	0.674	0.673
CBOW	0.701	0.675	0.672	0.670
Cognate	0.688	0.669	0.663	0.664

Table 4.19: Results for various edit distance algorithms with different numbers of neighbors.

For example, a language might have voicing assimilation of all consonants in a cluster. This would therefore mean that [t] and [d] would never appear next to the same consonants as each other and would thus have quite different distributions, despite only differing in voicing.

Furthermore, it may be the case that many of these methods are rendered of little use because the only possible edit distance neighbors being considered are the words that actually occur in the Portuguese corpus, which means that creating a more phonologically-informed model might not do much good because all candidates are already phonologically well-formed. For example, the feature-based edit distance algorithm would strongly indicate that *blanco* and *branco* are more likely to be cognates than *blanco* and *bkanco*; but there is no real need to make this distinction since no word like *bkanco* is going to occur in the Portuguese corpus anyway. If my aim were to generate Portuguese words given Spanish words, then these distinctions would be more important (since it would be crucial for the program to know not to generate *bkanco* from *blanco*), but since I am instead building a discriminative model, these distinctions may have less impact.

On the positive side, however, cognate-based embedding distance likely performs the best of the methods tested because its training most closely resembles the task upon which it is tested. That is, it is both trained and tested on the task of predicting cognates, whereas other algorithms use methods that only indirectly relate to the task of cognate determination.

Comparison of performance on known historical sound changes

The fundamental reason for trying all of these edit distance methods was to find the method that could most reliably identify cognates. The previous section quantitatively assesses success at this task, but it is not particularly easy to figure out how to interpret success at word alignment as a metric of edit distance success. Therefore, to give a less rigorous but easier to comprehend metric

of success, this section assesses how well each edit distance model performs at figuring out to assign low penalties to the character operations that correspond to common, systematic sound differences between Spanish and Portuguese.

Accordingly, I established a list of areas in which Spanish and Portuguese have systematic differences in how they have evolved from Latin; these differences are listed below and are derived from Boyd-Bowman (1980):

1. The Latin suffix *-arius* developed into *-ero* in Spanish but *-eiro* in Portuguese. Examples: Sp. *caldera* "boiler", Port. *caldeira* "boiler"; Sp. *primero* "first", Port. *primeiro* "first" (Boyd-Bowman 1980, 18-20)
2. The Latin suffix *-aticus* developed into Spanish *-aje* and Portuguese *-agem*. Examples: Sp. *viaje* "journey", Port. *viagem* "journey"; Sp. *coraje* "courage", Port. *coragem* "courage" (Boyd-Bowman 1980, 21-23)
3. In some non-word-initial positions, Latin <c> [k] became Portuguese <z> [z] but Spanish <c> [s]. Examples: Sp. *cocina* "kitchen", Port. *cozinha* "kitchen"; Sp. *placer* "pleasure", Port. *prazer* "preasure" (Boyd-Bowman 1980, 28-31)
4. Latin clusters of a consonant plus [l] became Spanish <j> [x] or Portuguese <lh> [ʎ]. Examples: Sp. *ojo* "eye", Port. *olho* "eye"; Sp. *lenteja* "lentil", Port. *lentilha* "lentil" (Boyd-Bowman 1980, 35)
5. Latin [kt] clusters became Spanish <ch> [tʃ] and Portuguese <it> [it]. Examples: Sp. *noche* "night", Port. *noite* "night"; Sp. *ocho* "eight", Port. *oito* "eight" (Boyd-Bowman 1980, 41)
6. Latin word-initial [f] became a now-silent <h> in Spanish but remained as <f> [f] in Portuguese. Examples: Sp. *haba* "broad bean", Port. *fava* "broad bean"; Sp. *hender* "to split", Port. *fender* "to crack" (Boyd-Bowman 1980, 62-63)
7. Portuguese lost intervocalic <l> [l] and <n> [n], but Spanish did not. Examples: Sp. *volar* "to fly", Port. *voar* "to fly"; Sp. *general* "general", Port. *geral* "general" (Boyd-Bowman 1980, 78)
8. In many instances, Portuguese removed instances of hiatus (often ones arising due to the aforementioned deletion of [l] or [n]) by merging consecutive vowels that were retained in Spanish. Examples: Span. *venir* "to come", Port. *vir* "to come"; Span. *leer* "to read", Port. *ler* "to read" (Boyd-Bowman 1980, 13)
9. Latin intervocalic [b] often became <v> [v] in Port but stayed in Spanish. Examples: Sp. *deber* "to owe", Port. *dever* "to owe"; Sp. *caballo* "horse", Port. *cavalo* "horse" (Boyd-Bowman 1980, 107-109)
10. Latin *pl*, *cl*, and *fl* developed into *ll* in Spanish and *ch* in Portuguese. Examples: Sp. *llover* "to rain", Port. *chover* "to rain"; Sp. *llamar* "to call", Port. *chamar* "to call" (Boyd-Bowman 1980, 110)

11. Latin suffix *-tas* became *-tad* or *-dad* in Spanish or *-tade* or *-dade* in Portuguese. Examples: Sp. *libertad* “freedom”, Port. *libertade* “freedom”; Sp. *bondad* “goodness”, Port. *bondade* “goodness” (Boyd-Bowman 1980, 136)
12. Latin *x* [ks] developed into Spanish *j* [x] and Portuguese *x* [ʃ]. Examples: Sp. *lujo* “luxury”, Port. *luxo* “luxury”; Sp. *fijar* “to fix”, Port. *fixar* “to fix” Boyd-Bowman (1980)[151]

Each of these sound differences has been converted into one or more character transformations in Table 4.20. For example, the correspondence between Spanish <ch> and Portuguese <it> is represented by two character operations, namely $\text{sub}(c,i)$ and $\text{sub}(h,t)$. The penalty for each of these operations is then also given for each edit distance algorithm, along with the mean and median penalties for all substitutions and all deletions (which are equivalent to insertions). Then, each operation for which the given method assigned a lower-than-median penalty to that operation was bolded; ideally, all of these operations would have bolded values because, given that all operations in the table are common differences between Portuguese and Spanish, all of them should receive low penalties. Of all the methods, the cognate-based method does best at assigning lower penalties to these operations; and several of the ones it misses are those that deal with digraphs, which makes sense since it was trained on single letters. Thus, even though the cognate-based method was not trained on Spanish-Portuguese cognates, it does seem to have learned some relevant pan-Romance characteristics by learning from French, Italian, and Portuguese.

4.6 Word alignment experiments

For each of the main distributions involved with the IBM Models (see Section 2.1.1 for descriptions of these models), namely translation probability, alignment probability, fertility, and distortion, I modified the relevant IBM model(s) to allow for the pivot language. Each of these modifications is described in this section. For all experiments, I tried both Levenstein edit distance (since it is the standard method) as well as the cognate-based edit distance that was found to be the highest-performing edit distance algorithm in the previous experiments. Portuguese remained the pivot language for all of these alignment model experiments. In each case, Portuguese parameters were found by using the mgiza word alignment software (Gao and Vogel 2008); for running this software, I used the conventional settings of 5 iterations of IBM Model 1, 5 iterations of the HMM (as a replacement for IBM Model 2), 3 iterations of IBM Model 3, and 3 iterations of IBM Model 4.

4.6.1 Translation probability

In order to incorporate translation probability into a pivot-based model, I define the translation probability $t(e|s)$ between English word e and Spanish word s as

$$t(e|s) = \arg \max_{p \in P} \frac{t(e|p)}{ed(s,p)} \quad (4.5)$$

where P is the set of all Portuguese words and $ed(s,p)$ is the edit distance between s and p .

Change(s)	Operation	Levenshtein	VC	Feature	CBOW	Cognate
8	del(a)	1	101	103	11.24	0.87
8, 11	del(e)	1	101	103	11.25	0.75
1, 8	del(i)	1	101	103	11.12	0.64
8	del(o)	1	101	103	11.24	0.87
8	del(u)	1	101	103	11.13	0.92
2	sub(j,g)	1	101	102	10.12	0.83
2	ins(m)	1	101	103	11.28	1.18
3	sub(c,z)	1	101	102	10.17	0.76
4	sub(j,l)	1	101	102	10.12	1.48
4	ins(h)	1	101	103	11.23	0.82
5	sub(c,i)	1	102	104	10.17	1.15
5	sub(h,t)	1	101	102	10.11	1.33
6	sub(h,f)	1	101	101	10.26	0.94
7	del(l)	1	101	103	11.23	0.89
7	del(n)	1	101	103	11.05	0.91
8	sub(b,v)	1	101	102	10.14	1.42
9	sub(l,c)	1	101	103	10.09	1.33
9	sub(l,h)	1	101	102	10.24	1.38
12	sub(j,x)	1	101	103	10.27	0.94
–	Mean substitution	1	101.7	102.7	10.23	1.18
–	Median substitution	1	101	102	10.19	1.21
–	Mean deletion	1	101	103	11.17	0.96
–	Median deletion	1	101	103	11.23	0.92

Table 4.20: Edit distances for systematic phonological differences between Spanish and Portuguese.

Equation 2.5 (reproduced here as 4.6) can then be used essentially unchanged to generate a sentence’s word alignments:

$$p(\mathbf{a}|\mathbf{s}, \mathbf{e}) = \prod_{i=0}^{l_e} t(e_j|s_{a(j)}) \quad (4.6)$$

where $a(j)$ is the index of the word in \mathbf{p} to which e_j is aligned. This definition encodes my assumption that the Portuguese words with the smallest edit distance from the Spanish word under consideration are most likely to have similar meanings to the Spanish word.

In the typical instantiation of IBM Model 1, the choice of which English word to align with Spanish word s is made by iterating through all English words in the English sentence and finding which has the greatest translation probability for s . This pivot-based formalism adds another layer to that iteration: Now, for each Spanish word s , the choice of which word to align with is made by iterating through all of s ’s closest Portuguese edit distance neighbors, and for each of those iterating through all English words in the English sentence, to find which pair of a Portuguese neighbor and an English word yields the highest translation probability, and then aligning with that English word.

Tables 4.21 and 4.22 summarize the results when translation probability is the sole quantity used to create the alignments, using Levenshtein edit distance and cognate-based edit distance (respectively). Note that, in the iterative process of training the IBM models, each successive model refines the translation probabilities of the one before it. Therefore, these tables show the results using the Portuguese translation probabilities derived from each different IBM model (or the HMM stage). In general, the higher the model was on which translation probabilities were trained, the better the pivot-based word alignment algorithm performed.

Some researchers have found effective neural-based methods for assigning translation probabilities (Yang et al. (2013), Tamura et al. (2014)). Therefore, in addition to the statistical methods for calculating translation probability, I also tested the use of this neural-based method. The method works as follows: Use a deep neural network with 2 hidden layers. To create the input, take pairs of an English word e and a Portuguese word p , and then concatenate the word embeddings of e and p to create an input vector for the network. I used GloVe embeddings for English (Pennington et al. 2014) and Polyglot embeddings for Portuguese (Al-Rfou et al. 2013). The output of the network is then a single cell, and during training this is set to be $t(e|p)$ as calculated using IBM Model 4. Although Yang et al. (2013) and Tamura et al. (2014) both succeeded in achieving greater success with their trained networks than with the IBM Model 4 on which the networks were trained due to the greater generalizability that comes from using word embeddings, this neural-based approach did not yield very fruitful results for me, even after experimenting with the parameters of the network. The neural-based results are also reported in Tables 4.21 and 4.22.

I also tested this system where, instead of taking the argmax of Equation 4.5, I summed across all possible Portuguese neighbors. This yielded results that were comparable to the current methods but slightly worse, so I stuck with the instantiation shown in 4.5.

4.6.2 Alignment probability

For the incorporation of alignment probability into this system, I made the assumption that Spanish syntax is identical to Portuguese syntax, and thus the reorderings that operate between Spanish and

Model	Precision	Recall	AER
IBM Model 1	0.379	0.335	0.644
HMM	0.447	0.394	0.581
IBM Model 3	0.461	0.407	0.568
IBM Model 4	0.470	0.413	0.561
Neural	0.122	0.073	0.909

Table 4.21: Alignment results based only on translation probabilities derived from various IBM models, using Levenshtein edit distance as the edit distance algorithm.

Model	Precision	Recall	AER
IBM Model 1	0.386	0.341	0.638
HMM	0.452	0.399	0.576
IBM Model 3	0.469	0.414	0.560
IBM Model 4	0.476	0.419	0.554
Neural	0.099	0.059	0.9236

Table 4.22: Alignment results based only on translation probabilities derived from various IBM models, using cognate-based edit distance as the edit distance algorithm.

English are the same as the reorderings that operate between Portuguese and English. Therefore, since the alignment probabilities depend only on word indices (and not on word identities), I use the mgiza-trained Portuguese-English alignment probabilities unchanged as the Spanish-English alignment probabilities. Equation 2.6 can then be used as in standard IBM Model 2 to generate word alignments.

The results of this test, using the alignment probabilities created by various components of the mgiza pipeline, are shown in Tables 4.23 and 4.24. As with the translation probabilities, performance was better the higher the model used for alignment probabilities.

4.6.3 Fertility

For fertility, the assumption was made that a given Spanish word will have the same fertility distribution as its cognates in Portuguese. Thus, fertility was incorporated as follows: For each Spanish word s , choose the Portuguese word p that is the most likely intermediary between s and

Model	Precision	Recall	AER
HMM	0.520	0.456	0.514
IBM Model 3	0.521	0.457	0.513
IBM Model 4	0.522	0.458	0.513

Table 4.23: Alignment results based on translation and alignment probabilities derived from various IBM models, using Levenshtein edit distance as the edit distance algorithm.

Model	Precision	Recall	AER
HMM	0.528	0.464	0.506
IBM Model 3	0.529	0.466	0.504
IBM Model 4	0.530	0.466	0.504

Table 4.24: Alignment results based on translation and alignment probabilities derived from various IBM models, using cognate-based edit distance as the edit distance algorithm.

Model	Precision	Recall	AER
IBM Model 3	0.493	0.404	0.556
IBM Model 4	0.493	0.405	0.555

Table 4.25: Alignment results based on translation probability, alignment probability, and fertility derived from various IBM models, using Levenshtein edit distance as the edit distance algorithm.

any English word in the English sentence. Then, choose the fertility of s by sampling from the fertility distribution of p . The test was run with the fertility distributions generated by both Model 3 and Model 4. In comparison to the performance with just translation and alignment probabilities, the introduction of fertility made performance decrease in all cases.

4.6.4 Distortion

As with alignment probability, the assumption was made for distortion probability that Spanish and Portuguese have identical syntax. In addition, although alignment probability is not at all lexically dependent, distortion probability (at least as enacted by mgiza) is somewhat lexically dependent—that is, it depends on the word class of the word under consideration. Therefore, in order to incorporate the mgiza-trained Portuguese-English distortion probabilities into my model, I also make the additional assumption that Spanish words tend to belong to the same word classes as their Portuguese cognates. With these assumptions, for each Spanish word under consideration I can use the exact distortion probability computed for its Portuguese edit distance neighbor and then use Equation 2.8 unmodified. The results from this approach are shown in Tables 4.27 and 4.28. Fertility is not included in these calculations because it makes the results worse. In addition, distortion can potentially be used as a replacement for alignment (since both model syntax in some way), so these tables show the results of considering distortion on its own vs. both distortion and alignment.

Model	Precision	Recall	AER
IBM Model 3	0.505	0.412	0.546
IBM Model 4	0.503	0.411	0.548

Table 4.26: Alignment results based on translation probability, alignment probability, and fertility derived from various IBM models, using cognate-based edit distance as the edit distance algorithm.

Model	Precision	Recall	AER
IBM Model 4	0.514	0.452	0.519
IBM Model 2 + 4	0.536	0.470	0.500

Table 4.27: Alignment results based on translation, alignment, and distortion probability derived from various IBM models, using Levenshtein edit distance as the edit distance algorithm.

Model	Precision	Recall	AER
IBM Model 4	0.530	0.467	0.504
IBM Model 2 + 4	0.547	0.481	0.488

Table 4.28: Alignment results based on translation, alignment, and distortion probability derived from various IBM models, using cognate-based edit distance as the edit distance algorithm.

4.6.5 Word alignment results

Results and discussion

Table 4.29 summarizes the results of the various word alignment tasks by indicating the best performance for each type of word alignment system. Of the four main parameters used here (translation probability, alignment probability, fertility, and distortion probability), all but fertility brought significant improvements to the overall performance of my alignment method. The fact that fertility did not improve performance may well be because the assumption that a given Portuguese word will have the same fertility distribution as its Spanish cognates is not necessarily true; for example, cognates can easily be different parts of speech, which can affect fertility. However, the success of the other three parameters suggests that the other assumptions being made are at least somewhat reasonable—that is, the success of translation probability suggests that edit distance can be useful tool for determining lexical items, while the fact that alignment and distortion probabilities were helpful suggests that Spanish and Portuguese have syntax that is similar enough to use the same word order assumptions for each of them.

Visualization of performance

Figures 4.1 through 4.4 show visualizations of the alignments generated with different parameters between the two sentences in one of the testing pairs. Figure 4.1 shows how translation probability is enough to get many individual words correct, but that the resulting alignments are distributed

Model	Precision	Recall	AER
Trans. prob.	0.476	0.419	0.554
Trans. & align. prob.	0.530	0.466	0.504
Trans. & align. prob. & fert.	0.503	0.411	0.548
Trans. & align. prob. & dist.	0.547	0.481	0.488

Table 4.29: Overall IBM model results

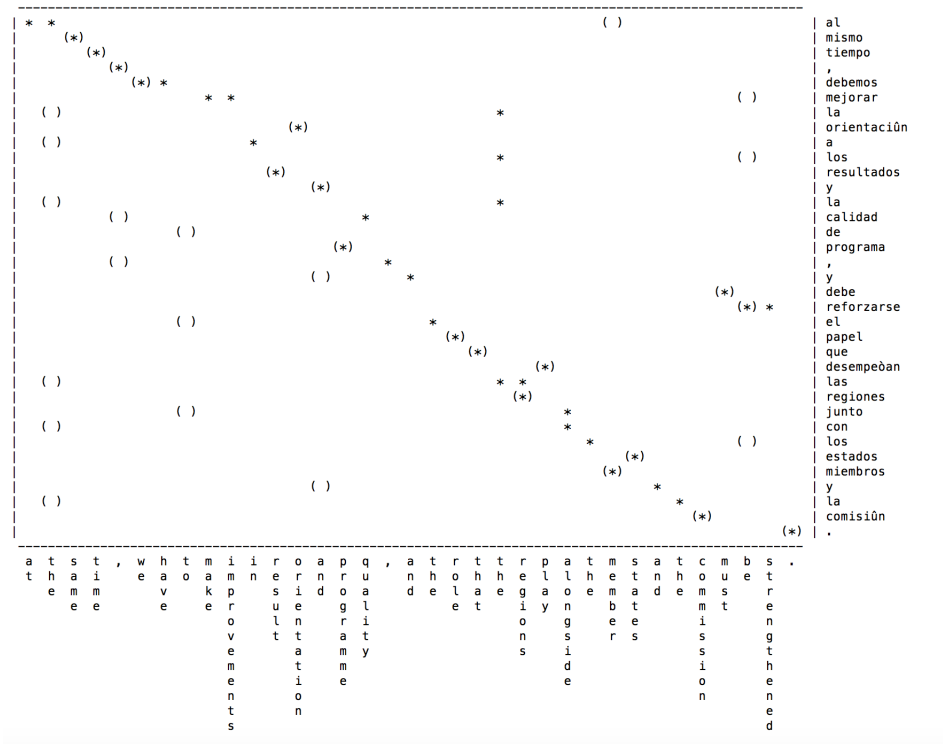


Figure 4.1: Visualization of a translation-probability-based alignment.

scattershot over the chart. Figure 4.2 shows much more order being imposed by the inclusion of alignment probability. Figure 4.3 shows how fertility allows some words to be aligned to a different number of words than 1. Finally, 4.4 shows how distortion probability brings similar effects as alignment probability, but it does enforce slightly more than alignment probability that there is little jumping occurring between alignments.

4.7 Pivot language experiments

The last major dimension of variation in my model is the choice of a pivot language. I tested 7 different pivot languages of varying relatedness to Spanish, namely Spanish itself, Portuguese, French, Italian, German, Danish, and Finnish. Figure 4.5 summarizes the relationships between these languages: Spanish is obviously most closely related to itself, then Portuguese is its next closest relative (as they are both in the Western Iberian group), then those two are joined by French in the Western Romance group, then Italian is next closest related within the entire Romance group, then German and Danish come next as other Indo-European languages, and finally Finnish is the outlier as a member of the Uralic languages, not known to be related to the Indo-European languages at all (Lewis et al. 2009).

We would expect that, the more closely related a language is to Spanish, the better it would serve as a pivot language. Table 4.30 shows the results of using various pivot languages; in all cases, the optimal word alignment parameters from the previous section were used (namely, factoring in

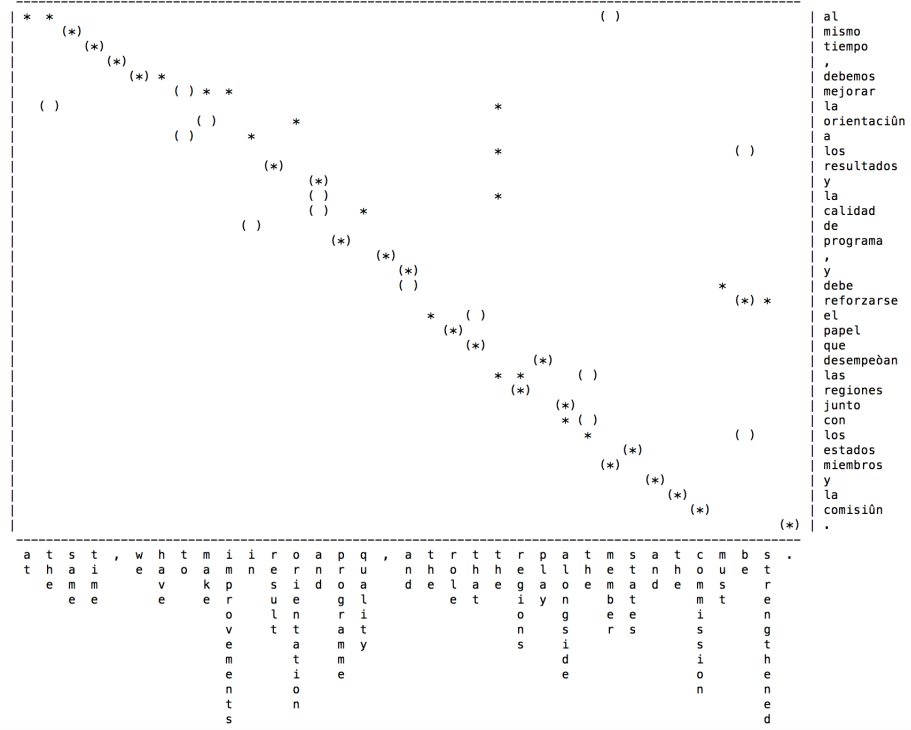


Figure 4.2: Visualization of a translation- and alignment-probability-based alignment.

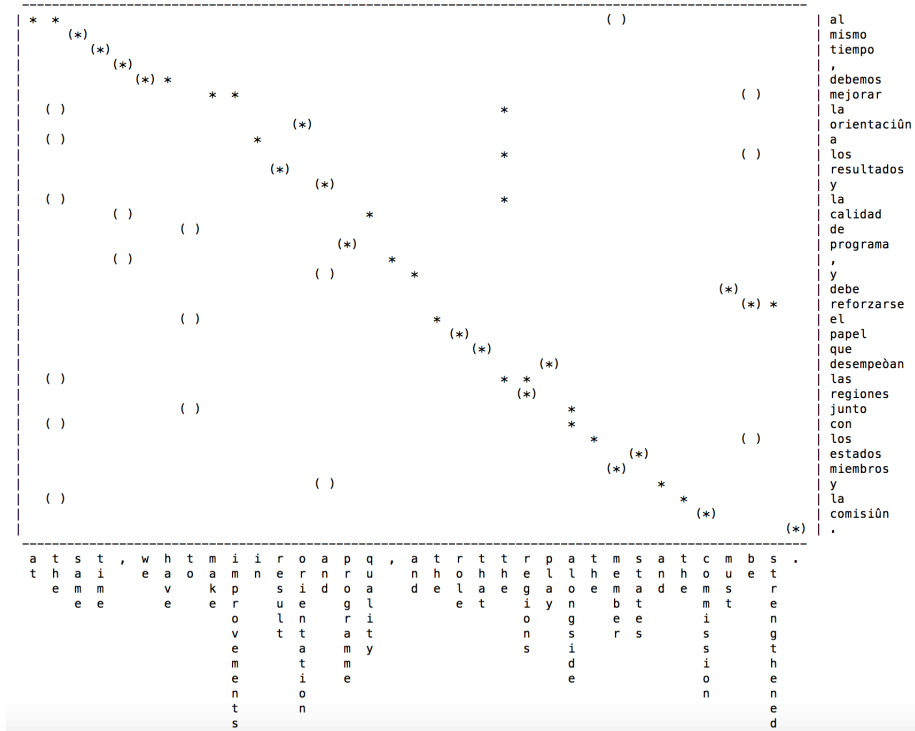


Figure 4.3: Visualization of a fertility-, translation- and alignment-probability-based alignment.

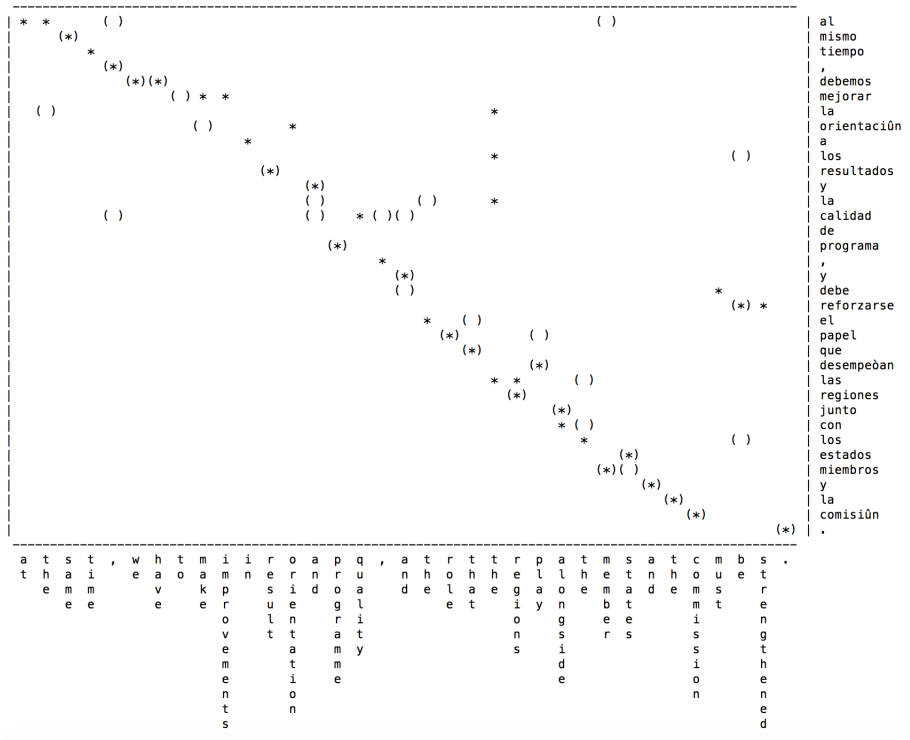


Figure 4.4: Visualization of a translation-, alignment-, and distortion-probability-based alignment.

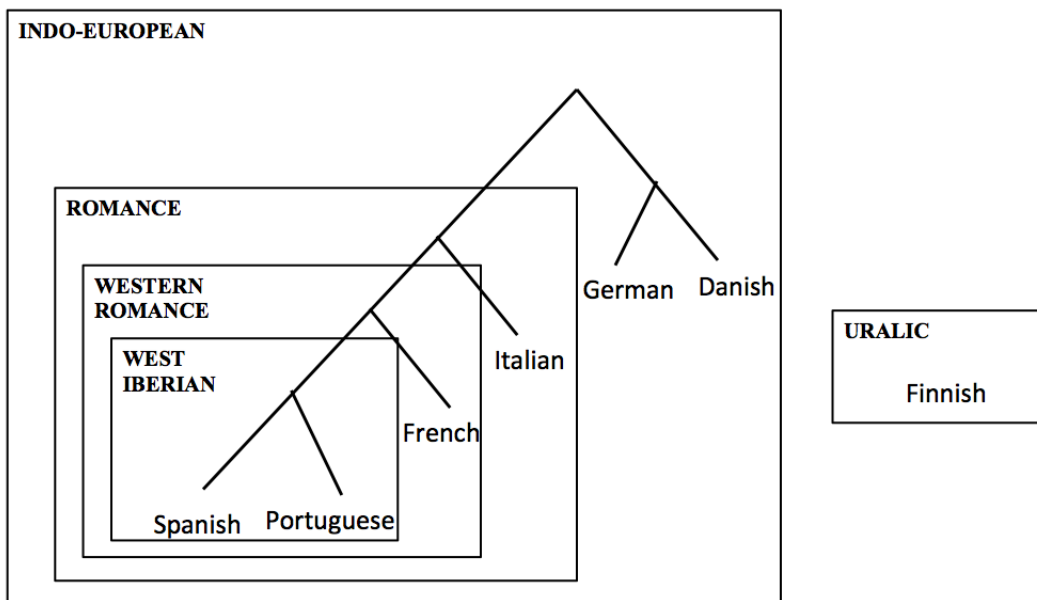


Figure 4.5: Family tree of the pivot languages used; language groups were derived from Ethnologue Lewis et al. (2009).

Language	Precision	Recall	AER
Spanish	0.682	0.602	0.360
Portuguese	0.536	0.470	0.500
Italian	0.458	0.402	0.571
French	0.416	0.361	0.613
German	0.300	0.259	0.722
Danish	0.289	0.251	0.732
Finnish	0.241	0.209	0.776

Table 4.30: Results from running the tests on various pivot languages.

translation probability, alignment probability, and distortion probability) while the edit distance algorithm used was always Levenshtein edit distance, which I used instead of cognate-based edit distance because it was unclear how best to deal with non-Romance languages in the realm of the Romance-based cognate edit distances. It is indeed generally the case that the performance of different pivot languages appears in the rank predicted by their relatedness within Figure 4.5. The only exception is that French and Italian are swapped in their ranking in Table 4.30 compared to their order predicted by Figure 4.5. One possible reason that the orders do not match perfectly is that historical relatedness of languages is not actually the factor most directly tied to utility as a pivot language; rather, the most important factor is lexical similarity. So, it may be the case that Italian just happens to have a slightly more similar set of words to Spanish than French does. Alternately, it may be that Italian syntax is more similar to Spanish syntax than French syntax is, or this discrepancy may simply have arisen from chance.

It is interesting to note that Finnish, despite being unrelated to Spanish, still performs significantly better than the lower-bound baselines. This is likely because there are some words, such as proper names and modern terms (such as the governmental terms that abound in this corpus), which tend to be the same or similar even across unrelated languages. In addition, the definition of *word* is quite liberal here and includes punctuation marks and numerals, which also tend to be similar across European languages at least. All in all, then, any pivot language seems to be better than no pivot language, but by far the best pivot language is one closely related to the low-resource language in question.

I also tried allowing the algorithm to consider edit distance neighbors from more than one other language. The theory behind this is that it might widen the net with which to catch a potential cognate. As an example, suppose that Latin had two words for “red”, namely red_1 and red_2 , and two words for “blue”, namely $blue_1$ and $blue_2$. Further, suppose that Spanish inherited only red_1 and $blue_1$, while Portuguese inherited only red_1 and $blue_2$, and French inherited only red_2 and $blue_1$. In this case, using both Portuguese and French would be beneficial because neither of them alone gives a full picture of Spanish but, together, they manage to triangulate both Spanish color words. In practice, however, any time that I allowed the aligner access to more than one pivot language, it performed worse than the better-performing pivot language would do on its own.

From the baseline tests, it is apparent that enforcing a diagonal alignment leads to great in-

Language	Precision	Recall	AER
Spanish	0.682	0.602	0.360
Portuguese	0.536	0.470	0.500
Italian	0.459	0.403	0.571
French	0.415	0.361	0.614
German	0.310	0.267	0.713
Danish	0.292	0.253	0.729
Finnish	0.255	0.221	0.763

Table 4.31: Results from running the tests with various pivot languages used for the translation probabilities but with Portuguese used for all alignment and distortion probabilities.

creases in performance even without any Spanish-specific knowledge added to the model. Therefore, although we showed in the previous section that factoring in the syntactic proxies or alignment and distortion probabilities did significantly improve performance, it is unclear if this is because the alignment and distortion probabilities have encoded specific syntactic information or if the improvement comes solely from the fact that the alignment and distortion probabilities are enforcing a diagonal alignment to some extent. To test this, I reran the tests in Table 4.30, but this time only the language used to derive translation probabilities changed, while Portuguese was used as the source of both the alignment and distortion probabilities in all cases. The results of these modified tests are in Table 4.31, and in general they do support the notion that the alignment and distortion probabilities are encoding at least some language-specific information, rather than just encoding a pan-language preference for diagonal alignments. This is evident because, in all the non-Romance languages (namely, German, Danish, and Finnish), substituting in Portuguese alignment and distortion probabilities decreased the AER, meaning that the Portuguese alignment and distortion values are better proxies for Spanish than German, Danish, and Finnish values. Meanwhile, the AERs for the Romance languages barely changed at all, suggesting that perhaps the alignment- and distortion-based improvements come from some general syntactic factors that are constant across Romance languages (meaning that any Romance language’s alignment and distortion values could capture these facts) but that are not captured in non-Romance languages (rendering the alignment and distortion values for other languages less useful). Thus, although most of the improvement from language to language does seem to be due to language-specific differences in the translation probabilities (for example, as Table 4.31 shows, Portuguese has an AER at least 0.07 better than all other languages even when the language used to derive the translation probabilities is the only variable), but there also do seem to be some improvements due to learned language-specific differences in syntax.

Chapter 5

Summary of results

5.1 Results

Results for each main category of tests are shown in Table 5.1. Overall, the best choices for the three parameters were as follows:

1. **Pivot language:** Portuguese
2. **Edit distance algorithm:** Cognate-based edit distance
3. **Alignment algorithm:** Translation probability, alignment probability, and distortion

Aligning with these parameters performed significantly better than the random and diagonal baselines, and it also came respectably close to the upper bounds on performance set by using Spanish as a pivot language and by aligning with cdec. In general, the following trends were found with respect to this data:

- Choice of edit distance algorithm does not have much effect on overall alignment success rate.
- Choice of alignment model is quite significant; each one of translation probability, alignment probability, and distortion probability improves the overall results by a noticeable amount.
- Choice of pivot language has a significant effect on the overall performance; specifically, the more closely-related the pivot language is to the low-resource language, the better the model will perform.

5.2 Example translations

The motivation for pursuing this algorithm for word alignment was the utility of word alignment in the task of machine translation; indeed, word alignment in and of itself does not accomplish much. Therefore, to illustrate the potential of this technique to be used in machine translation, I have

Edit distance algorithm	Alignment model	Pivot language	AER
-	Random	-	0.948
-	Diagonal	-	0.819
Levenshtein	IBM M1	Portuguese	0.673
VC	IBM M1	Portuguese	0.670
Feature-based	IBM M1	Portuguese	0.674
char2vec	IBM M1	Portuguese	0.672
Cognate-based	IBM M1	Portuguese	0.663
Cognate-based	IBM M1	Portuguese	0.554
Cognate-based	HMM	Portuguese	0.504
Cognate-based	IBM M3	Portuguese	0.548
Cognate-based	IBM M4	Portuguese	0.488
Levenshtein	IBM M4	Portuguese	0.500
Levenshtein	IBM M4	Italian	0.571
Levenshtein	IBM M4	French	0.613
Levenshtein	IBM M4	German	0.722
Levenshtein	IBM M4	Danish	0.732
Levenshtein	IBM M4	Finnish	0.776
Levenshtein	IBM M4	Spanish	0.360
-	fast-align	-	0.288

Table 5.1: General results.

combined my word alignment technique with the Moses package for statistical machine translation to create a rudimentary Spanish-English translation program.

The basic theory underlying this approach to translation is an application of Bayes' rule. Given a Spanish sentence \mathbf{s} , we want to find its most likely English translation \mathbf{e} . We could express this by saying that the translation into English is

$$\arg \max_{\mathbf{e}} p(\mathbf{e}|\mathbf{s})$$

Using Bayes' rule, we can transform this formula and apply the standard trick of removing the denominator (because \mathbf{s} is constant):

$$\arg \max_{\mathbf{e}} p(\mathbf{e}|\mathbf{s}) = \arg \max_{\mathbf{e}} \frac{p(\mathbf{s}|\mathbf{e})p(\mathbf{e})}{p(\mathbf{s})} \quad (5.1)$$

$$= \arg \max_{\mathbf{e}} p(\mathbf{s}|\mathbf{e})p(\mathbf{e}) \quad (5.2)$$

$p(\mathbf{s}|\mathbf{e})$ can be modeled with word-alignment-based probabilities, which have been established using my model. To round out the machine translation process, then, we just need to model $p(\mathbf{e})$, which Moses accomplishes through the use of an English trigram language model that it computes from the training data. Some examples of Spanish Europarl test sentences, with their true English translations and the English translations generated by this procedure, are given in Table 5.2.

Spanish: en el caso de turquía la cuestión es diferente .

Correct English: in the case of turkey it is different .

Initial translation: i like case of turkey de justice mr different .

Updated translation: in the case of turkey the question is different .

As this table makes apparent, this model often struggles with the translation of short words. The reason for this is that these translations depend heavily on using edit distance to find Portuguese cognates of the given Spanish words. A typical cognate will differ by a character or two from its version in the other language, and for long words a difference of a character or two is not very significant. However, for short words, changing a character or two can make the word unrecognizable, so essentially any word of the right length is a potential cognate of the word under consideration, and it is difficult to sift through all the short words to find the right one.

Thus far, I have been treating Spanish as essentially a no-resource language; that is, I have been assuming that we have zero Spanish data besides the Spanish sentence currently being aligned. However, in the real world such a scenario is extremely unlikely; even the lowest-resourced languages will typically have some data. As luck would have it, the sort of data that is available for a low-resource language is exactly the sort of data that the translations in Table 5.2 would benefit most from; that is, this translation model handles long words well but short words poorly, while the words for which it is easiest to glean data from a small data set are the most common words—which tend to be the short function words that this model struggles with. Therefore, I allowed myself the small luxury of treating Spanish as a low-resource language rather than a no-resource language by looking up a list of Spanish prepositions, articles, conjunctions, pronouns, and auxiliary verbs. All in all, there were a total of 65 Spanish words found for this purpose. I then incorporated these 65 Spanish words, along with their known translations, into the Moses model and gave high probabilities to

No.	Spanish	True English	Initial Translation	Updated Translation
1	a nuestro juicio esencial , ya que la propiedad intelectual es necesaria para estimular al mismo tiempo la innovación y la investigación .	in our view , it has a major part to play , as intellectual property is required in order to stimulate both innovation and research .	the essential i will , de , that intellectual property mr necessary to stimulate by the time de de innovation and research .	to our will , essential , that the intellectual property is necessary for stimulate by the time the innovation and the research .
2	en este debate voy a limitar mis observaciones a la cuestión de la coherencia y la vinculación entre la política regional y la política de competencia .	in this debate i will restrict my comments to the issue of coherence and the linkage between regional policy and competition policy .	i this debate to restrict the comments that the de justice of de coherence and de it between de regional policy and the policy of competence .	in this debate to to restrict have remarks to the question of the coherence and the link between the regional policy and the policy of competence .
3	por lo tanto , los únicos que viven realmente son los empresarios privados .	so really only private entrepreneurs are living .	for are both , for only that are really son for private entrepreneurs .	for both him , the only that are really are the entrepreneurs private .
4	señora presidenta , señora comisaria , el grupo del partido europeo de los liberales , demócratas y reformistas está enormemente preocupado por el cambio previsto del artículo 23 de la constitución de hong kong .	madam president , commissioner , the group of the european liberal , democrat and reform party is extremely concerned about the planned change to article 23 of hong kong 's constitution .	mr president , madam commissioner , like group of the european of for liberal , democrat and reform this enormously concerned for like exchange for of article 23 of the constitution of hong kong .	mr president , madam commissioner , the group of the of the european liberal , democrat and reform this enormously concerned for the exchange for of article 23 of the constitution of hong kong .
5	el número de refugiados que llegan al país es muy reducido en comparación con alemania , gran bretaña y otros países .	we have a very small number of refugees coming into the country , compared to germany or to britain and other countries .	like number of refugees that are by my country mr reduced i compared con germany , by britain and other countries .	the number of refugees that are by my country is reduced in compared with germany , by britain and other countries .
6	en el caso de turquía la cuestión es diferente .	in the case of turkey it is different .	i like case of turkey de justice mr different .	in the case of turkey the question is different .

Table 5.2: Example translations. The True English column shows the gold standard translation for the Spanish sentence. The Initial Translation column shows the translation before the Spanish function words were manually added to the Moses probability table. The Updated Translation show the translation after the Spanish function words were included.

their known translations. Simply adding in these 65 words noticeably improved the quality of the translations generated, as shown by the updated translations in Table 5.2.

Even in their updated versions, these translations are still far from perfect. Some of the mistakes made can be directly linked to the translation technique used; for example, in sentence 2, *competencia* is translated as *competence* rather than *competition* because Spanish *competencia* has a much smaller edit distance from the Portuguese word for “competence” (namely, *competência*) than the Portuguese word for “competition” (which is *concorrência*). In addition, in sentence 4, the translation has *mr president* where it should have *madam president*, likely because *mr president* is more common in the training data than *madam president* and thus gets rated more positively by the English language model. Despite these shortcomings, however, this translation method—which was trained on essentially no Spanish data and which was not tuned or optimized at all—does manage to roughly capture the meanings of the Spanish sentences it processes. The ability to create low-quality translations that nonetheless capture the meaning of a sentence could have some very real applications; for example, in a scenario in which a natural disaster strikes an area where a low-resource language is spoken, distress calls may start to be sent in the low-resource language, and a rapidly-generated rough translation of these distress calls could be all that is needed to inform the relief workers where to direct their efforts. This existing translator performs quite poorly on out-of-domain (i.e. non-political) text, but it can be supposed that a translator trained within the proper domain could carry out such disaster-necessitated translations, so these translation examples do therefore show that the technique of pivot-based word alignment used here can help tackle real-world problems.

Chapter 6

Conclusion

In this thesis, I have shown that pivot-based word alignment is a viable technique for creating word alignments that can be used to create rudimentary translations of novel text in a low-resource language for which the computer does not have access to any training data. This model makes use of the relationships between languages in the same language family, and the most successful parameters are the ones that display some sensitivity to historical sound changes between languages as well as language-specific lexical and syntactic information.

Throughout this paper, I have been assuming that there is no training data available at all, but the situation will rarely be that dire; thus, future work could focus on how to integrate cross-linguistic information when there are small (but non-zero) quantities of training data available. In addition, it may be fruitful to investigate how these techniques can be applied to phrase-based and/or neural machine translation to provide better translations than the rudimentary ones created thus far. Finally, even for the high-resource training languages in this project, I have not been using as much training data as may be necessary to achieve state-of-the-art word alignments, so further work could also embrace the big-data approach for the high-resource training to see if that further improves performance with the low-resource language.

Acknowledgements

Thank you to my suitemates for putting up with my strange hours while I worked on this thesis, to the other senior linguistics majors for much-needed moral support throughout the thesis-writing process, to Maria Piñango and Raffaella Zanuttini for guidance on how to prepare a thesis, to Drago Radev for agreeing to be my second reader, and most of all to Bob Frank for the many, many hours spent discussing the topics in this paper. All errors are, of course, my own.

Bibliography

- Al-Rfou, Rami, Bryan Perozzi and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the seventeenth conference on computational natural language learning*, 183–192. Sofia, Bulgaria: Association for Computational Linguistics.
- Bahdanau, Dzmitry, KyungHyun Cho and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR 2015)* .
- Boyd-Bowman, Peter. 1980. *From Latin to Romance in sound charts*. Georgetown University Press.
- Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19. 263–311.
- Chen, Stanley F. 1993. Aligning sentences in bilingual corpora using lexical information. *Proceedings of the 31st annual meeting on the Association for Computational Linguistics* 9–16.
- Cho, KyungHyun, Bart van Merriënboer, Dzmitry Bahdanau and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *International Conference on Learning Representations (ICLR 2013)* .
- Collobert, Ronan and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)* .
- Dyer, C., A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman and P. Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. *Proceedings of ACL* .
- Gale, William A. and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics* 19.1. 75–102.
- Gao, Qin and Stephan Vogel. 2008. Parallel implementations of word alignment tool. *Software engineering, testing, and quality assurance for natural language processing* 5. 49–57.
- Gerds, Donna B. and Lindsay Whaley. 1991. Locatives vs. instrumentals in Kinyarwanda. *Annual Meeting of the Berkeley Linguistics Society* 17.2.

- Koehn, Philipp, , Franz Josef Och and Daniel Marcu. 2003. Statistical phrase-based translation. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* 1. 48–54.
- Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. *MT Summit* 5. 79–86.
- Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. Mosesl open source toolkit for statistical machine translation. *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*) 177–180.
- Levenshtein, Vladimir I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8). 707–710.
- Lewis, M. Paul, Gary F. Simons and Charles D. Fennig. 2009. *Ethnologue: Languages of the world*, vol. 16. Dallas, TX: SIL International.
- Marcus, Mitchell P., Mary Ann Marcinkiewicz and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn treebank. *Computational linguistics* 19.2. 313–330.
- Mikolov, Tomas, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR 2013)* .
- Och, Franz Josef. 1999. An efficient method for determining bilingual word classes. *Proceedings of the ninth conference of the European chapter of the Association for Computational Linguistics* 71–76.
- Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics* 30.4. 417–449.
- Pennington, Jeffrey, Richard Socher and Christopher D. Manning. 2014. GloVe: Global vectors for word representation .
- Peperkamp, Sharon, Rozenn Le Calvez, Jean-Pierre Nadal and Emmanuel Dupoux. 2006. The acquisition of allophonic rules: Statistical learning with linguistic constraints. *Cognition* 101.3. B31–B41.
- Tamura, Akihiro, Taro Watanabe and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. *ACL* 1. 1470–80.
- Vogel, Stephan, Hermann Ney and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. *Proceedings of the 16th conference on computational linguistics* 2. 836–841.

Wagner, Robert and Michael J. Fischer. 1974. The string to string correction problem. *Journal of the ACM (JACM)* 21.1. 168–173.

Yang, Nan, Shujie Liu, Mu Li, Ming Zhou and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. *ACL* 1. 166–175.