Proceedings of the Society for Computation in Linguistics

Volume 1 Article 12

2018

Phonologically Informed Edit Distance Algorithms for Word Alignment with Low-Resource Languages

Richard T. McCoy
Johns Hopkins University, tom.mccoy@jhu.edu

Robert Frank
Yale University, robert.frank@yale.edu

Phonologically Informed Edit Distance Algorithms for Word Alignment with Low-Resource Languages

R. Thomas McCoy*
Department of Cognitive Science
Johns Hopkins University
tom.mccoy@jhu.edu

Robert Frank
Department of Linguistics
Yale University
robert.frank@yale.edu

Abstract

Edit distance is commonly used to relate cognates across languages. This technique is particularly relevant for the processing of low-resource languages because the sparse data from such a language can be significantly bolstered by connecting words in the low-resource language with cognates in a related, higher-resource language. We present three methods for weighting edit distance algorithms based on linguistic information. These methods base their penalties on (i) phonological features, (ii) distributional character embeddings, or (iii) differences between cognate words. We also introduce a novel method for evaluating edit distance through the task of low-resource word alignment by using edit-distance neighbors in a high-resource pivot language to inform alignments from the low-resource language. At this task, the cognate-based scheme outperforms our other methods and the Levenshtein edit distance baseline, showing that NLP applications can benefit from information about cross-linguistic phonological patterns.

1 Introduction

Many NLP techniques require large quantities of training data, which is a problem for low-resource languages (languages with little available data). Work on low-resource languages often focuses on tackling this low-data problem, such as by creating more data (Marton et al., 2009), collecting more data from the Internet (Mendels et al., 2015) or from scholarly papers (Xia et al., 2016), efficiently eliciting informative data (Probst et al., 2002), or crowdsourcing the collection of corpora (Post et al., 2012). One promising approach is to supplement the available data for a low-resource language with data from higher-resource languages, an approach which has been applied to tasks ranging from speech recognition (Thomas et al., 2012) to machine translation (Dholakia and Sarkar, 2014).

An open problem within this approach is finding the best way to map information from one language to another. When connecting related languages, a natural place to start is with cognates, and many works use edit distance for cognate detection (Simard et al., 1993; Barker and Sutcliffe, 2000; Koehn and Knight, 2000; Mann and Yarowsky, 2001; Inkpen et al., 2005; Bergsma and Kondrak, 2007; Munro and Manning, 2012). Edit distance refers to the difference between two strings, and this paper explores several techniques for determining edit distance. Our baseline is the Levenshtein edit distance algorithm (Levenshtein, 1966; Wagner and Fischer, 1974), and we introduce three novel edit distance algorithms, namely feature-based edit distance, char2vec-based edit distance, and cognate-based edit distance, and assess their performance at transferring information across languages using the task of low-resource cognate identification. Finally, we introduce a novel method for evaluating edit distance algorithms through the task of word alignment.

2 Related work

Ristad and Yianilos (1998) first presented schemes for training weighted edit distances, and others such

*Work done while at Yale University.

Proceedings of the Society for Computation in Linguistics (SCiL) 2018, pages 102-112.
Salt Lake City, Utah, January 4-7, 2018

as Cotterell et al. (2014) have proposed modifications to this method. Weighted edit distance and other weighted schemes for computing string similarity such as point-wise mutual information have been used by several authors for cognate detection (Kondrak, 2001; Ciobanu and Dinu, 2014; Jäger and Sofroniev, 2016; Jäger et al., 2017).

This paper's novel contribution is to extend this technique to a low-resource setting. The prior work in edit-distance-based cognate detection has relied on phonetic transcriptions, information about word meaning, or hand-created lists of cognates on which to train a system. Here we investigate how to assign edit distance weights when no such information is available for one of the languages in question. Several prior systems have used word similarity to assess historical linguistic claims about language phylogeny (Kondrak, 2002; Jäger, 2013; List, 2013), but here we follow the inverse strategy of using knowledge about language phylogeny to inform the determination of string similarity by compensating for the lack of resources about a language with information from closely related and better-resourced languages. An additional novel contribution of this paper is to propose a new technique for assessing string similarity metrics based on how much a given metric can improve performance on a practical NLP task.

3 Edit distance algorithms

We use four basic approaches to calculating edit distance, detailed in the following subsections.

3.1 Levenshtein edit distance

As a baseline, we use Levenshtein edit distance (Levenshtein, 1966; Wagner and Fischer, 1974). Levenshtein edit distance focuses on three 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 defined as the minimum number of insertions and/or deletions and/or substitutions that must be made to transform w_1 into w_2. Table 1 contains some examples of word pairs and the Levenshtein edit distance ($dist_L(w_1, w_2)$) between them.

3.2 Feature-based edit distance

This section details two approaches to edit distance in which the basic aim is to alter the Levenshtein penalties 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. Indeed, Kondrak (2002) showed that the ALINE system, which computes string similarity based on a sophisticated set of phonological features, performed better at cognate identification than the basic Levenshtein method did; however, this success does not necessarily extend to our current situation because, due to our assumption that we are in a low-resource environment, we use only orthographic representations of words, not phonetic transcriptions.

3.2.1 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 2.

3.2.2 More features

For this model, we assign a set of phonological features to each character and make the penalty for any operation equal to the number of features that change when that operation occurs. For example, substituting a d for a t incurs a penalty of 1 because a

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)
jim carrey	john kerry	6	sub(i,o), insert(h), sub(m, n), sub(c, k), sub(a, e), del(e)

Table 1: Examples of Levenshtein edit distance.

single feature (namely, voicing) has changed, while substituting a b for a t incurs a penalty of 2 because two features (voicing and place) have changed. In practice, it is impossible to enact this approach rigorously because orthography does not map cleanly to phonology and does not have consistent phonological properties across languages. Therefore, many of the weights used for this method are by necessity somewhat arbitrary because the set of phonological features that we assign to each character does not necessarily match that character's true features in all contexts and across all languages.

3.3 Char2vec edit distance

The word2vec algorithm from Mikolov et al. (2013a) uses word distributions to train representations of words as vectors in high-dimensional vector space; such vector representations are called embeddings. Inspired by word2vec's success at creating semantically sensible embeddings for words, we apply the word2vec algorithm to characters in an attempt to create phonologically sensible embeddings for characters. We refer to this technique as *char2vec*. The char2vec algorithm begins by considering a window of a fixed size around every instance of character c in the training corpus (which, in this case, was the monolingual Portuguese data from the Europarl corpus; see Section 4 for more details). We tested windows of size 3 and 5. For word2vec, larger windows are typically used, but since there are far fewer characters than 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, so there is little need to look farther away from the target character to populate the cooccurrence vector.

Once the desired windows around different occurrences of c were established, a neural network was used to generate the vector embedding for c. The neural network used was 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 was then trained using either the continuous bag of words (CBOW) method or the skip-gram method, both described in Mikolov et al. (2013a). Once the network finished training, the trained weight matrix used to transition from the input layer to the hidden layer was used to generate the embeddings for all of the characters. Specifically, for the character at index i in the input vector, its embedding was row i of the weight matrix. An embedding was also created for the empty string ϵ by pretending that there was an ϵ between every two letters in the training data.

Once these embeddings were trained, the char2vec edit distance between any two characters was defined as one divided by the cosine distance between the embeddings for those two characters, which is given by the following equation:

$$(cosdist(c_1, c_2))^{-1} = \frac{||\vec{c_1}||\ ||\vec{c_2}||}{\vec{c_1} \cdot \vec{c_2}} \qquad (1)$$

where $\vec{c_1}$ and $\vec{c_2}$ are the vector embeddings of c_1 and c_2. The negative exponent is there because the cosine is greater for more similar vectors, whereas we want a smaller penalty for more similar vectors. For insertions and deletions, the same equation is used, except that either c_1 (for insertions) or c_2 (for deletions) is ϵ because deleting a character can be thought of as replacing it with the empty string, and inserting a character can be thought of as replacing the empty string with that character.

These embedding-based edit distances are founded upon two assumptions: First, as with the feature-based edit distance methods in Section 3.2, these methods assume that, across a pair of cognates, it is more likely for one character to substitute 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, it is not necessarily the case that distributional evidence is useful for cognate determination, since identifying allophones within a language might entail significantly different types of evidence than identifying cognates across languages.

3.4 Cognate-based edit distance

The embedding-based methods in the previous section all derive their embeddings from a single training language (in this case Portuguese). We now try to utilize cross-linguistic information from three Romance languages, namely Portuguese, Italian, and French. 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 Romance cognate pairs and which are not and to apply this information to the test language of Spanish (which is not used as a training language).

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 (1), (2), (3), and (4a), while negative examples were identified by finding any pairs (w_1, w_2) that satisfied criteria (1), (2), (3), and (4b):

1. w_1 and w_2 are from different languages.
2. The Levenshtein edit distance between w_1 and w_2 is greater than 0 but less than some specified amount d.
3. Both w_1 and w_2 are at least 4 characters long.
4. (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 (Pennington et al., 2014) of the most likely English translation of w_1 and w_2 is less than 0.5.

For criterion (1), the languages considered were Portuguese, French, and Italian, which are all of the Romance languages (besides the test language of Spanish) considered in this paper. For criterion (2), we ran the experiments both with $d = 1$ and with $d = 2$. Criterion (3) 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 (4), the most likely English translation of a word is identified based on the IBM Model 1 (Brown et al., 1993) translation probabilities generated by running the mgiza word alignment program (Gao and Vogel, 2008) on the bilingual Portuguese/English, French/English, and Italian/English training sets. Finally, for criterion (4b), we 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 (4a) is meant to find positive examples by identifying words that appear phonologically similar and have similar meanings.

When d from criterion (2) 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. We 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 2 shows some of the positive and negative example pairs generated when $d = 1$.

These examples were used to train weights for each possible operation of insertion, deletion, or substitution. There are 27 characters for which to learn weights (the 26 letter plus an OTHER character[1]); 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

[1] A character with a diacritic is represented as 2 characters, the base character plus a diacritic character that is collapsed into the OTHER category.

Word 1	Word 2	Word 1	Word 2
afgane (It.) “afghan”	afghane (Fr.) “afghan”	colmando (It.) “closing”	comando (Port.) “command”
stupide (Fr.) “stupid”	stupido (It.) “stupid”	eternamente (It.) “eternally”	externamente (Port.) “externally”
serviu (Port.) “served”	servi (Fr.) “served”	monge (Port.) “monk”	ronge (Fr.) “plaguing”
discriminata (It.) “against”	discriminada (Port.) “against”	paute (Port.) “transparent”	faute (Fr.) “fault”
finali (It.) “final”	finale (Fr.) “final”	mentis (It.) “mindset”	mentir (Fr.) “lie”

Table 2: 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).

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. We used logistic regression to find the weights that performed best at classifying the training items as cognates or non-cognates.

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 not necessarily true; a language pair could easily 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.

4 Experimental setups

For all experiments, Spanish is treated as if it is a low-resource language for which we wish to gain information based on its high-resource relatives of Portuguese, Italian, and French. Although Spanish is a very high-resource language in real life, for these experiments we simulate low-resourcedness by not providing the computer with any Spanish training data; thus, the test data is the computer's only exposure to Spanish. We chose this path rather than using a truly low-resource language because it is much easier to create gold standards for evaluation for a high-resource language. As in real life, Portuguese, Italian, and French are treated as high-resource (that is, there is ample training data for these languages), as is English, which is used in the word alignment experiments. Thus, the char2vec embeddings are trained on Portuguese data, and the cognate-based edit distances are trained on Portuguese, French, and Italian data. All experiments used the Europarl parallel corpus (Koehn, 2005) as sources of text in the languages of interest. This corpus comes from the proceedings of the European Parliament, a governing body of the European Union.

4.1 Cognate identification

Cognate identification was used as a direct method of testing how well each edit distance algorithm performed. To do this, a set of pairs of likely Spanish/Portuguese cognates was formed in the same way as the likely cognates were chosen for the cognate-based embeddings (see Section 3.4). Call such a Spanish/Portuguese pair (s, p). For each such pair, each edit distance algorithm was used to identify the Portuguese word $p_{closest}$ with the smallest edit distance from s (with ties being broken randomly). It was then checked whether $p = p_{closest}$. A total of 12,198 cognate pairs were tested in this manner for each edit distance algorithm.

4.2 Word alignment

4.2.1 Background

Though cognate identification is the most direct method for evaluating each edit distance algorithm, it has several flaws. First, it requires the selection of exactly one Portuguese word as the cognate for a given Spanish word, when in reality there may

be many valid Portuguese choices (such as other inflected forms of the intended Portuguese word). More generally, cognacy only really makes sense at the lemma level, but the low-resource setting of our task means that we do not have access to lemmatization and must use words instead of lemmas, making the cognate detection task ill-defined in this context. In addition, the fact that the method of selecting cognates so closely mirrors the steps for training cognate-based edit distance may unfairly advantage the cognate-based edit distance algorithm[2]. Therefore, as a fairer and more definitely quantifiable assessment, we use the task of 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. Word alignment is an important step in many machine translation systems, such as the popular Moses software system (Koehn et al., 2007), and effective word alignment depends heavily upon the size of the alignment algorithm's training corpus. Therefore, success at word alignment (and, by extension, machine translation programs based on word alignment) suffers greatly under a data shortage. Cognate information can be used to combat this data shortage because, if a low-resource language is related to a high-resource language, educated guesses about the meanings of words in the low-resource language may be formed based on similar-looking words in the high-resource language. Presumably, an edit distance algorithm that more accurately identifies cognates will perform better at this sort of pivoting than an edit distance algorithm that does not perform as well at cognate identification. Multiple authors in the past have worked on exploiting language relatedness to assist in machine translation (Mikolov et al., 2013b; Zoph et al., 2016; Cheng et al., 2016), including via a focus on using this information to improve word alignment performance (Xiang et al., 2010).

4.2.2 Pivot-based alignment algorithm

All word alignment experiments aim to align Spanish sentences with their English translations, with no Spanish-English training data available. Instead, a bilingual Portuguese-English training set of 1 million lowercased and tokenized sentences from the Europarl corpus is used to train Portuguese-English translation probabilities of the form $t(e|p)$, where $t(e|p)$ is the probability that a given Portuguese word p will be translated as the English word e. The training of these translation probabilities was accomplished using mgiza (Gao and Vogel, 2008), which is an implementation of the IBM models of word alignment (Brown et al., 1993). The test set for the word alignment experiments is a set of 1,000 lowercased and tokenized parallel Spanish/English sentences from the Europarl corpus. 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, but for the purposes of low-resource NLP these errors are expected to be negligible.

In order to pivot from Portuguese (the pivot language used for training) to Spanish (the low-resource language used for testing), we define the translation score[3] $t(e|s)$ between English word e and Spanish word s as

$$t(e|s) = \arg\max_{p \in P} \frac{t(e|p)}{ed(s,p) + \lambda n} \qquad (2)$$

where P is the set of all Portuguese words, $ed(s,p)$ is the edit distance between s and p, and λn is the product of a smoothing factor λ and the number of edit operations n, where λ was optimized for each edit distance algorithm to scale the relative importance of t and ed. This definition encodes our assumption that similar-looking Spanish and Portuguese words (i.e., Spanish and Portuguese words with low edit distances) will have similar meanings.

The Spanish and English sentences are then aligned based solely on this translation score (as in IBM Model 1), without reference to any of the properties such as distortion or fertility used in higher IBM models. The choice to only use translation

[2] An anonymous reviewer notes that this second problem could be overcome by testing on human-generated cognate lists, which would be a useful metric to compute in future work. However, the other problem with the cognate task remains.

[3] We call it a score rather than a probability because we do not normalize, so the scores do not sum to 1.

probability was made for simplicity; since the focus of these experiments is on edit distance algorithms, not on alignment algorithms, it was simplest to use the most basic alignment algorithm.

In the typical instantiation of IBM Model 1, the choice of which target word to align with source word a is made by iterating through all words in the target sentence and finding which has the greatest translation probability for a. This pivot-based formalism adds another step: 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 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. Because IBM Model 1 treats all word alignments as independent, the probability of a set **a** of word alignments that align English sentence **e** and Spanish sentence **s** can be maximized simply by maximizing the probabilities of each individual alignment between a Spanish word s and some English word e—that is, by aligning each Spanish word s with the English word e that maximizes $t(e|s)$.

5 Evaluation

An advantage of the word alignment task is that we can straightforwardly quantify the results via the Alignment Error Rate (AER), defined as

$$\text{AER} = 1 - \frac{2P}{p + c} \tag{3}$$

where P is the number of predicted alignments that are correct, c is the number of alignments in the gold standard, and p is the number of predicted alignments (where an alignment is defined as a connection between one Spanish word and one English word). AER falls within the range of 0 to 1, where it is best to be as close to 0 as possible.

6 Results and discussion

6.1 Cognate identification

The results at the cognate identification task are reported in Table 3. The cognate-based results are reported with $d = 2$, where d is the maximum edit distance between French/Italian, French/Portuguese, and Italian/Portuguese cognate pairs used to train the weights for edit distance operations, since $d = 2$ performed better than $d = 1$. This is likely because increasing the maximum edit distance between cognate pairs creates more pairs for the training set.

Algorithm	Accuracy
Levenshtein	0.314
Vowel-Consonant	0.316
More Features	0.320
Char2vec	0.312
Cognate	**0.330**

Table 3: Results on cognate identification.

6.2 Word alignment

Table 4 shows the results at the word alignment task. The first set of methods are included as baselines. Random refers to randomly aligning each Spanish word with one English word, while Diagonal refers to aligning the i^{th} Spanish word with the i^{th} English word for all i less than the minimum of the two sentences' lengths, so these two methods show how well an extremely naive model can perform. Meanwhile, fast-align is a state-of-the-art word alignment program from the cdec package (Dyer et al., 2010) and was trained directly on the Spanish-English section of Europarl; thus, its performance is indicative of the best performance that can reasonably be expected on this task in a high-resource setting.

The second part of the table compares the various edit distance algorithms. To scale each algorithm's set of penalties to a reasonable range of penalty ratios, all algorithms were tested with each smoothing factor λ in the set [0.01, 1, 5, 10, 50, 100], where the smoothing factor in question was added to each penalty used in the calculation of a word pair's overall edit distance. Results are reported with the best-performing smoothing factor for each algorithm.

The char2vec algorithm was tested both using the CBOW and the skip-gram methods from Mikolov et al. (2013a) as well as with window sizes of 3 and 5 (i.e., 1 character on either side of the target word and 2 characters on either side of the target word). CBOW and skip-gram performed comparably; because CBOW performed slightly better, results are reported with it. The window size of 3 performed significantly better than a window size of 5, so re-

Edit distance algorithm	λ	Alignment model	Pivot langauge	AER
-	-	Random	-	0.948
-	-	Diagonal	-	0.819
-	-	fast-align	-	0.288
Levenshtein	0.01	IBM M1	Portuguese	0.673
VC	100	IBM M1	Portuguese	0.670
More features	100	IBM M1	Portuguese	0.674
Char2vec	10	IBM M1	Portuguese	0.672
Cognate	1	IBM M1	Portuguese	**0.663**
Cognate	1	IBM M1	Portuguese	0.554
Cognate	1	HMM	Portuguese	0.504
Cognate	1	IBM M3	Portuguese	0.548
Cognate	1	IBM M4	Portuguese	**0.448**
Levenshtein	0.01	IBM M4	Spanish	0.360
Levenshtein	0.01	IBM M4	Portuguese	0.500
Levenshtein	0.01	IBM M4	Italian	0.571
Levenshtein	0.01	IBM M4	French	0.613
Levenshtein	0.01	IBM M4	German	0.722
Levenshtein	0.01	IBM M4	Danish	0.732
Levenshtein	0.01	IBM M4	Finnish	0.776

Table 4: Smoothed word alignment results for various experimental settings.

sults are reported with this window size. This difference is likely because there are so few characters in the alphabet that considering characters in a wider window ceases to uniquely characterize a given character, since pretty much any character can easily occur two letters away from pretty much any other character. For the cognate-based edit distance, as in Section 6.1, results are reported with $d = 2$.

Part 2 of this table only uses IBM Model 1. To see whether more advanced models can improve performance, the third section of Table 4 uses basic implementations of the higher IBM models from Brown et al. (1993) and the HMM model from Vogel et al. (1996). For each of these models, the parameters from training on Portuguese-English alignment were transferred directly to the Spanish-English case. Note that, in this section, IBM M1 really refers to using the IBM Model 1 algorithm with translation probabilities trained using IBM Model 4; this is why the IBM Model 1 performs better in part 3 of the table than in part 2, because part 2 only uses translation probabilities trained with IBM Model 1.

Finally, the fourth part of Table 4 shows results with various pivot languages that vary in their level of relatedness to Spanish. Spanish itself is included in this section as a baseline for the best possible performance under the pivot-based framework.

6.3 Discussion

For word alignment, feature-based edit distance did not beat the Levenshtein baseline, while the vowel-consonant based edit distance led to a modest improvement. These results may arise from the fact that the only 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* will occur in the Portuguese corpus anyway.

The char2vec method also led to modest improvements; its performance may have been hindered by its assumption that distributional similarity implies phonological similarity, when in fact there are some reasons to suppose the contrary. For example, a language might have voicing assimilation of all conso-

nants in a cluster. This would 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.

Cognate-based edit distance had the strongest performance. Since it was only trained on cognate pairs using Romance languages other than Spanish, while it was tested on Spanish words, this result justifies our assumption that facts about the phonological relatedness of Spanish's relatives can also be used to learn useful information about Spanish. The cognate identification task corroborates the word alignment results in indicating that cognate-based edit distance is the best-performing algorithm for bootstrapping information from a high-resource language to one of its low-resource relatives.

The second part of Table 4 shows that three more linguistically informed algorithms presented here (particularly cognate-based edit distance) outperform the less-informed basic Levenshtein algorithm. These results broadly suggest that incorporating linguistic information can be of significant benefit to NLP applications with low-resource languages, since it was helpful here to utilize information about the phonological relatedness between languages rather than using the flat distribution of the basic Levenshtein algorithm. Though the alignment results in the second part of Table 4 are not impressive at an absolute level, the results in the third part of the table show that alignment performance can be significantly improved by preserving the same pivot-based setup but using more advanced alignment algorithms, so there is hope that refining the alignment algorithms more could further improve performance. Finally, the bottom segment of Table 4 shows that alignment performance generally improves as the pivot language becomes more closely related to Spanish, corroborating the claim that it is language relatedness that fuels the success of the pivot-based alignment method. (Figure 1 shows a family tree of the pivot languages used.)

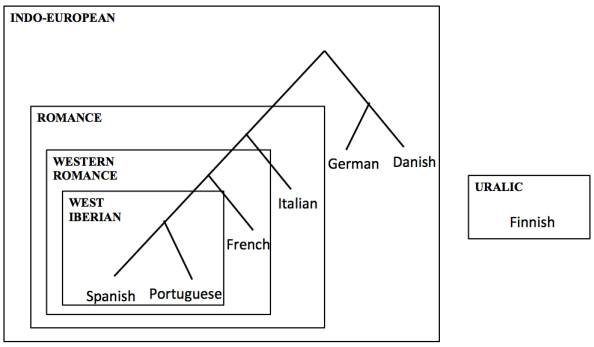

Figure 1: Family tree of the pivot languages used; language groups were derived from Ethnologue (Lewis et al., 2009).

7 Conclusions and future work

We have presented three new techniques for computing edit distance. All of these make use of more linguistic information (specifically, cross-linguistic phonological information) than the baseline of Levenshtein edit distance, and all of them perform at least as well as Levenshtein edit distance at low-resource cognate identification and word alignment. In particular, cognate-based edit distance brings the greatest performance improvements in these tasks compared to Levenshtein edit distance. This work focuses on the IBM alignment models, so future work could explore more advanced algorithms relating to word alignment and machine translation, such as neural machine translation (Collobert and Weston, 2008; Cho et al., 2014; Bahdanau et al., 2015), or phrase-based machine translation (Koehn et al., 2003; Och and Ney, 2004) to bring alignment performance improvement. In addition, the edit distance algorithms could be refined to encode more phonological information. For example, separate penalties could be assigned based on the environment in which changes occur since environment is highly significant in phonological changes both within and across languages. Another refinement specific to the cognate-based algorithm would be training on a list of likely cognates to choose weights and then using those weights to choose an updated list of likely cognates, and iterating this process until the weights converge; the algorithm as presented here only represents one iteration of such a process, but further iterations might yield better weights.

Acknowledgments

We would like to thank the anonymous reviewers and the members of Yale's 2017 Senior Essay in Linguistics class for their helpful suggestions on this work. Any errors remain our own.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR 2015)*.

Gosia Barker and Richard FE Sutcliffe. 2000. An experiment in the semi-automatic identification of false-cognates between english and polish. *Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science*.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. *Annual meeting-Association for Computational Linguistics*, 45.

Peter F. Brown, 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.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1604.02201*.

KyungHyun Cho, 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)*.

Alina Maria Ciobanu and Liviu P. Dinu. 2014. Automatic detection of cognates using orthographic alignment. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, volume 2: Short Papers, ACL 2014,*, pages 99–105.

Ronan Collobert 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)*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic fsts. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, volume 2: Short Papers, ACL 2014,*, pages 625–630.

Rohit Dholakia and Anoop Sarkar. 2014. Pivot-based triangulation for low-resource languages. *Proc. AMTA*.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. *Proceedings of the ACL 2010 System Demonstrations*.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. *Software engineering, testing, and quality assurance for natural language processing*, 5:49–57.

Diana Inkpen, Oana Frunza, and Grzegorz Kondrak. 2005. Automatic identification of cognates and false friends in french and english. *Proceedings of the International Conference Recent Advances in Natural Language Processing*, 9.

Gerhard Jäger and Pavel Sofroniev. 2016. Automatic cognate classification with a support vector machine. *Proceedings of the 13th Conference on Natural Language Processing*, 16:128–134.

Gerhard Jäger, Johann-Mattis List, and Pavel Sofroniev. 2017. Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1204–1215.

Gerhard Jäger. 2013. Phylogenetic inference from word lists using weighted alignment with empirically determined weights. *Language Dynamics and Change 3*, 2:245–291.

Philipp Koehn and Kevin Knight. 2000. Estimating word translation probabilities from unrelated monolingual corpora using the em algorithm. *AAAI/IAAI*.

Philipp Koehn, 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.

Philipp Koehn, 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. Moses open source toolkit for statistical machine translation. *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions)*, pages 177–180.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. *MT Summit*, 5:79–86.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, pages 1–8.

Grzegorz Kondrak. 2002. *Algorithms for language reconstruction*. Ph.D. thesis, University of Toronto.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

M. Paul Lewis, Gary F. Simons, and Charles D. Fennig. 2009. *Ethnologue: Languages of the world*, volume 16. SIL International, Dallas, TX.

Johann-Mattis List. 2013. *Sequence comparison in historical linguistics*. Ph.D. thesis, Heinrich-Heine-Universität Düsseldorf.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved statistical machine translation using monolingually-derived paraphrases. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 1.

Gideon Mendels, Erica Cooper, Victor Soto, Julia Hirschberg, Mark JF Gales, Kate M. Knill, Anton Ragni, and Haipeng Wang. 2015. Improving speech recognition and keyword search for low resource languages using web data. *Sixteenth Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR 2013)*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Robert Munro and Christopher D. Manning. 2012. Accurate unsupervised joint named-entity extraction from unaligned parallel text. *Proceedings of the 4th Named Entity Workshop*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30.4:417–449.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation.

Sharon Peperkamp, 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.

Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six indian languages via crowdsourcing. *Proceedings of the Seventh Workshop on Statistical Machine Translation*.

Katharina Probst, Jaime Carbonell, and Lori Levin. 2002. Semi-automatic learning of transfer rules for machine translation of low-density languages. *Proceedings of the Student Session at the 14th European Summer School in Logic, Language and Information (ESSLLI-02)*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 5:522–532.

Michel Simard, George F. Foster, and Pierre Isabelle. 1993. Using cognates to align sentences in bilingual corpora. *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing*, 2.

Samuel Thomas, Sriram Ganapathy, and Hynek Hermansky. 2012. Multilingual mlp features for low-resource lvcsr systems. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. *Proceedings of the 16th conference on computational linguistics*, 2:836–841.

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

Fei Xia, William D. Lewis, Michael Wayne Goodman, Glenn Slayden, Ryan Georgi, Joshua Crowgey, , and Emily M. Bender. 2016. Enriching a massively multilingual database of interlinear glossed text. *Language Resources and Evaluation*, 50:321–349.

Bing Xiang, Yonggang Deng, and Bowen Zhou. 2010. Diversify and combine: Improving word alignment for machine translation on low-resource languages. *Proceedings of the ACL 2010 Conference Short Papers*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.