Orthographic Features for Bilingual Lexicon Induction

Parker Riley and **Daniel Gildea**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

Recent embedding-based methods in bilingual lexicon induction show good results, but do not take advantage of orthographic features, such as edit distance, which can be helpful for pairs of related languages. This work extends embedding-based methods to incorporate these features, resulting in significant accuracy gains for related languages.

1 Introduction

Over the past few years, new methods for bilingual lexicon induction have been proposed that are applicable to low-resource language pairs, for which very little sentence-aligned parallel data is available. Parallel data can be very expensive to create, so methods that require less of it or that can utilize more readily available data are desirable.

One prevalent strategy involves creating multilingual word embeddings, where each language's vocabulary is embedded in the same latent space (Vulić and Moens, 2013; Mikolov et al., 2013a; Artetxe et al., 2016); however, many of these methods still require a strong cross-lingual signal in the form of a large seed dictionary.

More recent work has focused on reducing that constraint. Vulić and Moens (2016) and Vulic and Korhonen (2016) use document-aligned data to learn bilingual embeddings instead of a seed dictionary. Artetxe et al. (2017) use a very small, automatically-generated seed lexicon of identical numerals as the initialization in an iterative self-learning framework to learn a linear mapping between monolingual embedding spaces; Zhang et al. (2017) use an adversarial training method to learn a similar mapping. Lample et al. (2018a) use a series of techniques to align monolingual embedding spaces in a completely unsupervised way; their method is used by Lample et al. (2018b) as the initialization for a completely unsupervised machine translation system.

These recent advances in unsupervised bilingual lexicon induction show promise for use in low-resource contexts. However, none of them make use of linguistic features of the languages themselves (with the arguable exception of syntactic/semantic information encoded in the word embeddings). This is in contrast to work that predates many of these embedding-based methods that leveraged linguistic features such as edit distance and orthographic similarity: Dyer et al. (2011) and Berg-Kirkpatrick et al. (2010) investigate using linguistic features for word alignment, and Haghighi et al. (2008) use linguistic features for unsupervised bilingual lexicon induction. These features can help identify words with common ancestry (such as the English-Italian pair *agile-agile*) and borrowed words (*macaroni-maccheroni*).

The addition of linguistic features led to increased performance in these earlier models, especially for related languages, yet these features have not been applied to more modern methods. In this work, we extend the modern embedding-based approach of Artetxe et al. (2017) with orthographic information in order to leverage similarities between related languages for increased accuracy in bilingual lexicon induction.

2 Background

This work is directly based on the work of Artetxe et al. (2017). Following their work, let $X \in \mathbb{R}^{|V_s| \times d}$ and $Z \in \mathbb{R}^{|V_t| \times d}$ be the word embedding matrices of two distinct languages, referred to respectively as the source and target, such that each row corresponds to the d-dimensional embedding of a single word. We refer to the ith row of one of

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 390–394
Melbourne, Australia, July 15 - 20, 2018.

these matrices as X_{i*} or Z_{i*}. The vocabularies for each language are V_s and V_t, respectively. Also let $D \in \{0,1\}^{|V_s| \times |V_t|}$ be a binary matrix representing a dictionary such that $D_{ij} = 1$ if the ith word in the source language is aligned with the jth word in the target language. We wish to find a mapping matrix $W \in \mathbb{R}^{d \times d}$ that maps source embeddings onto their aligned target embeddings. Artetxe et al. (2017) define the optimal mapping matrix W^* with the following equation,

$$W^* = \arg\min_W \sum_i \sum_j D_{ij} \|X_{i*}W - Z_{j*}\|^2$$

which minimizes the sum of the squared Euclidean distances between mapped source embeddings and their aligned target embeddings.

By normalizing and mean-centering X and Z, and enforcing that W be an orthogonal matrix ($W^T W = I$), the above formulation becomes equivalent to maximizing the dot product between the mapped source embeddings and target embeddings, such that

$$W^* = \arg\max_W \text{Tr}(XWZ^T D^T)$$

where $\text{Tr}(\cdot)$ is the trace operator, the sum of all diagonal entries. The optimal solution to this equation is $W^* = UV^T$, where $X^T DZ = U\Sigma V^T$ is the singular value decomposition of $X^T DZ$.

This formulation requires a seed dictionary. To reduce the need for a large seed dictionary, Artetxe et al. (2017) propose an iterative, self-learning framework that determines W as above, uses it to calculate a new dictionary D, and then iterates until convergence. In the dictionary induction step, they set $D_{ij} = 1$ if $j = \arg\max_k (X_{i*}W) \cdot Z_{k*}$ and $D_{ij} = 0$ otherwise.

We propose two methods for extending this system using orthographic information, described in the following two sections.

3 Orthographic Extension of Word Embeddings

This method augments the embeddings for all words in both languages before using them in the self-learning framework of Artetxe et al. (2017). To do this, we append to each word's embedding a vector of length equal to the size of the union of the two languages' alphabets. Each position in this vector corresponds to a single letter, and its value is set to the count of that letter within the spelling of the word. This letter count vector is then scaled by a constant before being appended to the base word embedding. After appending, the resulting augmented vector is normalized to have magnitude 1.

Mathematically, let A be an ordered set of characters (an alphabet), containing all characters appearing in both language's alphabets:

$$A = A_{source} \cup A_{target}$$

Let O_{source} and O_{target} be the orthographic extension matrices for each language, containing counts of the characters appearing in each word w_i, scaled by a constant factor c_e:

$$O_{ij} = c_e \cdot \text{count}(A_j, w_i), O \in \{O_{source}, O_{target}\}$$

Then, we concatenate the embedding matrices and extension matrices:

$$X' = [X; O_{source}], \quad Z' = [Z; O_{target}]$$

Finally, in the normalized embedding matrices X'' and Z'', each row has magnitude 1:

$$X''_{i*} = \frac{X'_{i*}}{\|X'_{i*}\|}, \quad Z''_{i*} = \frac{Z'_{i*}}{\|Z'_{i*}\|}$$

These new matrices are used in place of X and Z in the self-learning process.

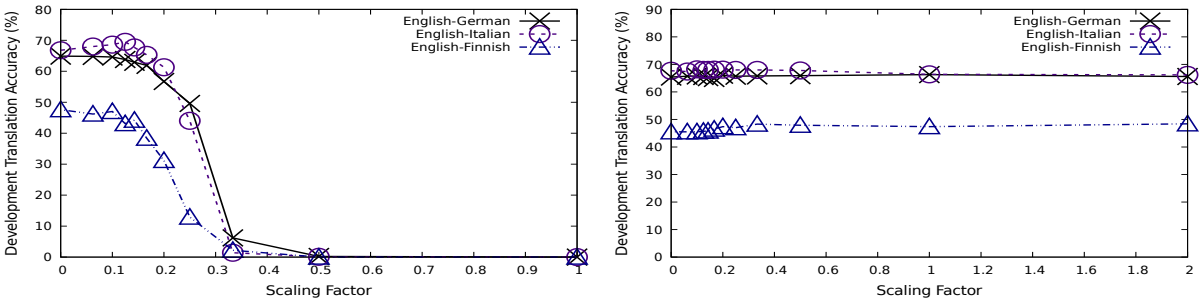
4 Orthographic Similarity Adjustment

This method modifies the similarity score for each word pair during the dictionary induction phase of the self-learning framework of Artetxe et al. (2017), which uses the dot product of two words' embeddings to quantify similarity. We modify this similarity score by adding a measure of orthographic similarity, which is a function of the normalized string edit distance of the two words.

The normalized edit distance is defined as the Levenshtein distance ($\text{L}(\cdot,\cdot)$) (Levenshtein, 1966) divided by the length of the longer word. The Levenshtein distance represents the minimum number of insertions, deletions, and substitutions required to transform one word into the other. The normalized edit distance function is denoted as $\text{NL}(\cdot,\cdot)$.

$$\text{NL}(w_1, w_2) = \frac{\text{L}(w_1, w_2)}{\max(|w_1|, |w_2|)}$$

We define the orthographic similarity of two words w_1 and w_2 as $\log(2.0 - \text{NL}(w_1, w_2))$. These



(a) Embedding Extension

(b) Similarity Adjustment

Figure 1: Performance on development data vs. scaling factors c_e and c_s. The lowest tested value for both was 10^{-6}.

similarity scores are used to form an orthographic similarity matrix S, where each entry corresponds to a source-target word pair. Each entry is first scaled by a constant factor c_s. This matrix is added to the standard similarity matrix, XWZ^T.

$$S_{ij} = c_s \cdot \log(2.0 - \mathrm{NL}(w_i, w_j)), w_i \in V_s, w_j \in V_t$$

The vocabulary for each language is 200,000 words, so computing a similarity score for each pair would involve 40 billion edit distance calculations. Also, the vast majority of word pairs are orthographically very dissimilar, resulting in a normalized edit distance close to 1 and an orthographic similarity close to 0, having little to no effect on the overall estimated similarity. Therefore, we only calculate the edit distance for a subset of possible word pairs.

Thus, the actual orthographic similarity matrix that we use is as follows:

$$S'_{ij} = \begin{cases} S_{ij} & \langle w_i, w_j \rangle \in \texttt{symDelete}(V_t, V_s, k) \\ 0 & \text{otherwise} \end{cases}$$

This subset of word pairs was chosen using an adaptation of the Symmetric Delete spelling correction algorithm described by Garbe (2012), which we denote as `symDelete(·,·,·)`. This algorithm takes as arguments the target vocabulary, source vocabulary, and a constant k, and identifies all source-target word pairs that are identical after k or fewer deletions from each word; that is, all pairs where each is reachable from the other with no more than k insertions and k deletions. For example, the Italian-English pair *moderno-modern* will be identified with $k = 1$, and the pair *tollerante-tolerant* will be identified with $k = 2$.

The algorithm works by computing all strings formed by k or fewer deletions from each target word, stores them in a hash table, then does the same for each source word and generates source-target pairs that share an entry in the hash table. The complexity of this algorithm can be expressed as $O(|V|l^k)$, where $V = V_t \cup V_s$ is the combined vocabulary and l is the length of the longest word in V. This is linear with respect to the vocabulary size, as opposed to the quadratic complexity required for computing the entire matrix. However, the algorithm is sensitive to both word length and the choice of k. In our experiments, we found that ignoring all words of length greater than 30 allowed the algorithm to complete very quickly while skipping less than 0.1% of the data. We also used small values of k ($0 < k < 4$), and used $k = 1$ for our final results, finding no significant benefit from using a larger value.

5 Experiments

We use the datasets used by Artetxe et al. (2017), consisting of three language pairs: English-Italian, English-German, and English-Finnish. The English-Italian dataset was introduced in Dinu and Baroni (2014); the other datasets were created by Artetxe et al. (2017). Each dataset includes monolingual word embeddings (trained with word2vec (Mikolov et al., 2013b)) for both languages and a bilingual dictionary, separated into a training and test set. We do not use the training set as the input dictionary to the system, instead using an automatically-generated dictionary consisting only of numeral identity translations (such as 2-2, 3-3, et cetera) as in Artetxe et al. (2017).[1] However, because the methods presented in this work feature tunable hyperparameters, we use a portion of the training set as devel-

[1] https://github.com/artetxem/vecmap

Method	English-German	English-Italian	English-Finnish
Artetxe et al. (2017)	40.27	39.40	26.47
Artetxe et al. (2017) + identity	51.73	44.07	42.63
Embedding extension, $c_e = \frac{1}{8}$	50.33	48.40	29.63
Embedding extension + identity, $c_e = \frac{1}{8}$	55.40	47.13	**43.54**
Similarity adjustment, $c_s = 1$	43.73	39.93	28.16
Similarity adjustment + identity, $c_s = 1$	52.20	44.27	41.99
Combined, $c_e = \frac{1}{8}$, $c_s = 1$	53.53	**49.13**	32.51
Combined + identity, $c_e = \frac{1}{8}$, $c_s = 1$	**55.53**	46.27	41.78

Table 1: Comparison of methods on test data. Scaling constants c_e and c_s were selected based on performance on development data over all three language pairs. The last two rows report the results of using both methods together.

Source Word	Our Prediction (Language)	Incorrect Baseline Prediction (Translation)
caesium	*cäsium* (German)	*isotope* (*isotope*)
unevenly	*ungleichmäßig* (German)	*gleichmäßig* (*evenly*)
Ethiopians	*Äthiopier* (German)	*Afrikaner* (*Africans*)
autumn	*autunno* (Italian)	*primavera* (*spring*)
Brueghel	*Bruegel* (Italian)	*Dürer* (*Dürer*)
Latvians	*latvialaiset* (Finnish)	*ukrainalaiset* (*Ukrainians*)

Table 2: Examples of pairs correctly identified by our embedding extension method that were incorrectly translated by the system of Artetxe et al. (2017). Our system can disambiguate semantic clusters created by word2vec.

opment data.[2] In all experiments, a single target word is predicted for each source word, and full points are awarded if it is one of the listed correct translations. On average, the number of translations for each source (non-English) word was 1.2 for English-Italian, 1.3 for English-German, and 1.4 for English-Finnish.

6 Results and Discussion

For our experiments with orthographic extension of word embeddings, each embedding was extended by the size of the union of the alphabets of both languages. The size of this union was 199 for English-Italian, 200 for English-German, and 287 for English-Finnish.

These numbers are perhaps unintuitively high. However, the corpora include many other characters, including diacritical markings and various symbols (%, [, !, etc.) that are an indication that tokenization of the data could be improved. We did not filter these characters in this work.

For our experiments with orthographic similarity adjustment, the heuristic identified approximately 2 million word pairs for each language pair out of a possible 40 billion, resulting in significant computation savings.

[2]We use all source-target pairs containing one of 1,000 randomly-selected target words.

Figure 1 shows the results on the development data. Based on these results, we selected $c_e = \frac{1}{8}$ and $c_s = 1$ as our hyperparameters. The local optima were not identical for all three languages, but we felt that these values struck the best compromise among them.

Table 1 compares our methods against the system of Artetxe et al. (2017), using scaling factors selected based on development data results. Because approximately 20% of source-target pairs in the dictionary were identical, we also extended all systems to guess the identity translation if the source word appeared in the target vocabulary. This improved accuracy in most cases, with some exceptions for English-Italian. We also experimented with both methods together, and found that this was the best of the settings that did not include the identity translation component; with the identity component included, however, the embedding extension method alone was best for English-Finnish. The fact that Finnish is the only language here that is not in the Indo-European family (and has fewer words borrowed from English or its ancestors) may explain why the performance trends for English-Finnish were different than those of the other two language pairs.

In addition to identifying orthographically similar words, the extension method is capable of

learning a mapping between source and target *letters*, which could partially explain its improved performance over our edit distance method.

Table 2 shows some correct translations from our system that were missed by the baseline.

7 Conclusion and Future Work

In this work, we presented two techniques (which can be combined) for improving embedding-based bilingual lexicon induction for related languages using orthographic information and no parallel data, allowing their use with low-resource language pairs. These methods increased accuracy in our experiments, with both the combined and embedding extension methods providing significant gains over the baseline system.

In the future, we want to extend this work to related languages with different alphabets (experimenting with transliteration or phonetic transcription) and to extend other unsupervised bilingual lexicon induction systems, such as that of Lample et al. (2018a).

Acknowledgments We are grateful to the anonymous reviewers for suggesting useful additions. This research was supported by NSF grant number 1449828.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 2289–2294, Austin, Texas.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 451–462, Vancouver, Canada.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the 2010 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-10)*, pages 582–590.

Georgiana Dinu and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *CoRR*, abs/1412.6568.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*, pages 409–419.

Wolf Garbe. 2012. 1000x faster spelling correction algorithm. `http://blog.faroo.com/2012/06/07/improved-edit-distance-based-spelling-correction/`. Accessed: 2018-02-12.

Aria Haghighi, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 771–779.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Ivan Vulic and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 247–257, Berlin, Germany.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, pages 1613–1624, Seattle, Washington, USA.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *J. Artif. Int. Res.*, 55(1):953–994.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-17)*, pages 1959–1970, Vancouver, Canada.