Using Word Vectors to Improve Word Alignments for Low Resource Machine Translation

Nima Pourdamghani, Marjan Ghazvininejad, Kevin Knight
Information Sciences Institute & Department of Computer Science
University of Southern California
{damghani,ghazvini,knight}@isi.edu

Abstract

We present a method for improving word alignments using word similarities. This method is based on encouraging common alignment links between semantically similar words. We use word vectors trained on monolingual data to estimate similarity. Our experiments on translating fifteen languages into English show consistent BLEU score improvements across the languages.

1 Introduction

Word alignments are essential for statistical machine translation (MT), especially in low-resource settings where neural MT systems often do not compete with phrase-based and syntax-based MT (Koehn and Knowles, 2017). The most widely used word alignment method (Brown et al., 1993) works by estimating the parameters of IBM models from training data using the Expectation Maximization (EM) algorithm. However, EM works poorly for low-frequency words as they do not appear enough in the training data for confident parameter estimation. This problem is even worse in low-resource settings where a large portion of word types appear infrequently in the parallel data. In this paper we improve word alignments and consequently machine translation in low resource settings by improving the alignments of infrequent tokens.

Works that deal with the rare-word problem in word alignment include those that alter the probability distribution of IBM models' parameters by adding prior distributions (Vaswani et al., 2012; Mermer and Saraçlar, 2011), smoothing the probabilities (Moore, 2004; Zhang and Chiang, 2014; Van Bui and Le, 2016) or introducing symmetrization (Liang et al., 2006; Pourdamghani et al., 2014). These works, although effective, merely rely on the information extracted from the parallel data. Another branch adds linguistic knowledge like word stems, orthography (Hermjakob, 2009) morphological analysis (De Gispert et al., 2006; Lee, 2004), syntactic constraints (Fossum et al., 2008; Cherry and Lin, 2006; Toutanova et al., 2002) or a mixture of such clues (Tiedemann, 2003). These methods need language-specific knowledge or tools like morphological analyzers or syntax parsers that is costly and time consuming to obtain for any given language.

A less explored branch that can help aligning rare words is adding semantic information. The motivation behind this branch is simple: Words with similar meanings should have similar translations. Previously, Ma et al. (2011) cluster words using monolingual data and substitute each word with its cluster representative to get alignments. They then duplicate their parallel data and use both regular alignments and alignments on word classes for training MT. Kočiskỳ et al. (2014) simultaneously learn alignments and word representations from bilingual data. Their method does not benefit from monolingual data and requires large parallel data for training. Songyot and Chiang (2014) define a word-similarity model that can be trained from monolingual data using a feed-forward neural network, and alter the implementation of IBM models in Giza++ (Och and Ney, 2003) to use the word similarity inside their EM. They require large monolingual data for both source language and English. While English monolingual data is abundant, availability of large and reliable monolingual data for many low resource languages is not guaranteed.

All these previous works define their own word similarity models, which similar to the more widely used distributed word representation methods (Mikolov et al., 2013; Pennington et al., 2014), assign high similarity to substitutable words in a given context; however, substitutability does not

Proceedings of NAACL-HLT 2018, pages 524–528
New Orleans, Louisiana, June 1 - 6, 2018.

always imply synonymy. For instance *tea* and *coffee*, or *Pakistan* and *Afghanistan* will be similar in these models but do not share translations.

In this paper we propose a simple method to use off-the-shelf distributed representation methods to improve word alignments for low-resource machine translation (Section 2). Our model is based on encouraging common alignment links between semantically similar words. We do this by extracting a bilingual lexicon, as a subset of the translation tables trained by IBM models and adding it to the parallel data. For instance, the rare word *obliterated* and its semantically similar word *destroyed*, have a common entry *destruida* in the English/Spanish translation table. We add a new (*obliterated*, *destruida*) pair to the parallel data to encourage aligning *obliterated* to *destruida*.

The simplicity of our method makes it easy to be widely used. Our work addresses a major problem of previous works, which is taking substitutability for synonymy without discrimination. Finally, the lexicon can be extracted either with or without help of word vectors trained on foreign language monolingual data. Large and reliable foreign monolingual data can help our alignments, but we still get good improvements over baseline for languages with small monolingual data where we only use English word vectors (Section 4).

We test our method on both alignment f-score and machine translation BLEU (Section 4). Alignment accuracy is tested on Arabic-English, Chinese-English and Farsi-English gold alignments. Machine translation accuracy is tested on fifteen languages were we show a consistent BLEU score improvement.

2 Proposed Method

We improve the alignment of rare words by encouraging them to align to what their semantic neighbors align to. For instance we encourage the rare word *obliterated* to align to what *destroyed* aligns to. However, we should be careful in this process. Distributed word representation methods like (Mikolov et al., 2013; Pennington et al., 2014) often define word similarity as the ability to substitute one word for another given a context. This does not always imply having same translations. Multiple reasons contribute to this problem. First, word vectors are noisy, especially when monolingual data is small. Second, some words might have multiple meanings and a semantically similar word might share only part of these meanings. Finally, some words do not have synonyms, especially proper names. Word vectors often group such entities together as they are substitutable, but this similarity should not be used for alignments.

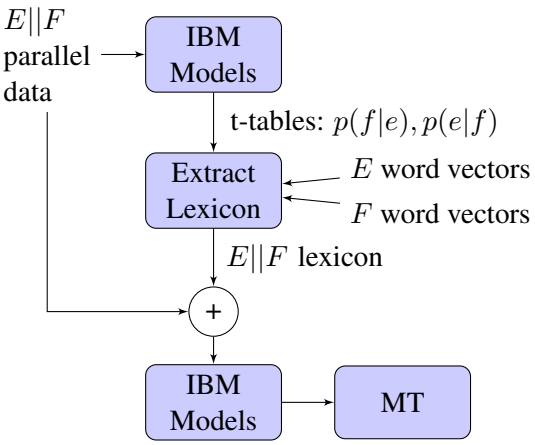

Figure 1: Word vectors trained on monolingual data are used to extract a bilingual lexicon out of translation tables. This lexicon is added to the parallel data, resulting in improved alignments for machine translation.

We bring a simple three-fold solution to these problems. First, we split the use of English and foreign word vectors in the method, so that if foreign monolingual data is small or unreliable, we can fall back to only using English word vectors. Second, and more importantly, we limit the effect of a semantic neighbor on the alignments of a token to the common alignment links between them. This removes the effect of a semantic neighbor which is not a synonym (like effect of *tea* on alignments of *coffee*) and irrelevant meanings of a semantic neighbor (like multiple meanings of *bow* on alignments of token *crossbow*) as we only encourage an alignment link if it appears as a potential translation for both the neighbors. Third, we note that using similarities based on distributed representations only hurts alignments for proper names. For these cases we encourage alignment to transliterations if applicable.

Figure 1 shows the outline of the proposed method. We provide the initial parallel data to the IBM models and train the translate tables $p(f|e)$ and $p(e|f)$. We then use the word vectors trained on English and foreign language monolingual data to extract a bilingual lexicon from these tables. This lexicon is added to the original parallel data and used to re-train the alignments. The lexicon contains both common alignment links and transliteration links that are extracted from the

translation table. Next we will describe how each section of the lexicon is generated.

2.1 Extracting Semantically Similar Tokens

Assume an infrequent English token e (w.r.t. the parallel data), and its semantic neighbor e'. If e and e' have a common t-table entry—some f, where $p(f|e) > 0$ and $p(f|e') > 0.1$ we encourage the translation of e to f by adding this pair to the parallel data for re-alignment. We limit the lexicon entries to non-common words. We only add entries where $freq(e) \leq 100$ and $freq(f) \leq 100$. The translation table is trained by 5 iterations of each of IBM models 1, HMM, and 4.

We add each (e,f) pair multiple times to the lexicon proportional to $p(f|e')$, the cosine distance of e and e', and the frequency of e. More precisely, for each neighbor e', each (e, f) pair appears $\lceil min(freq(e) \times dist(e, e') \times p(f|e'), \frac{freq(e)}{4}) \rceil$ times in the lexicon.

To measure similarity, we use cosine distance of word vectors trained on monolingual data using an implementation of continues bag-of-words (CBOW) algorithm.[1] English word vectors are trained on the one-billion-word language modeling benchmark (Chelba et al., 2013). Foreign language word vectors are trained on the monolingual data described in Section 3. All vectors are trained with window size 6 and dimension 300. For each word we consider its two nearest neighbors according to the cosine distance.

In a similar manner, we extract a lexicon from the $p(e|f)$ translation table as well. For each foreign rare token f and its semantic neighbor f', we add (e, f) pair to the lexicon if $p(e|f) > 0$ and $p(e|f') > 0.1$. However, as discussed in Section 4, it is better to use this lexicon only if the foreign language word vectors are trained on more than 10 million tokens of monolingual data.

2.2 Extracting Transliterations

For any infrequent English token e (w.r.t. the parallel data) and its translation table entry f, if f is a transliteration of e we add the (e, f) pair to the lexicon. Similarly we extract transliteration pairs from the $p(e|f)$ translation table. Each transliteration pair is added once to the lexicon.

In order to decide whether two tokens are transliterations, we compute the normalized edit distance of their romanizations. We use uroman,[2] a universal romanizer that converts text in any script to its romanized version (Latin alphabet). We say two tokens are transliterations if $dist(rom(e), rom(f)) \leq 0.25$, where $dist$ is the normalized Levenshtein distance and $rom(.)$ is the output of the romanizer.

3 Data

We use data from fifteen languages for our machine translation experiments.[3] These languages include Amheric, Arabic, Bengali, Mandarin, Farsi, Hausa, Somali, Spanish, Tamil, Thai, Turkish, Uighur, Urdu, Uzbek and Yoruba. Table 1 shows the size of training, development, test and monolingual data for each language. In addition, we use hand aligned data[4] for Arabic/English ($77.3K$+$119.5K$ tokens), Chinese/English ($240.2K$+$305.2K$ tokens), and Farsi/English ($0.9K$+$0.8K$ tokens) for word alignment experiments. We lowercase and tokenize all data using Moses (Koehn et al., 2007) scripts.

	train	dev.	test	mono.
amh	2.1M	39.8K	19.5K	4.3M
ara	3.8M	39.1K	19.8K	230.4M
ben	0.9M	41.9K	21.0K	2.5M
cmn	10.6M	41.7K	20.5K	33.2M
fas	4.3M	47.7K	24.2K	271.2M
hau	2.1M	48.0K	24.1K	3.9M
som	2.8M	46.8K	23.5K	13.5M
spa	24.1M	49.4K	24.3K	14.7M
tam	0.5M	39.0K	11.4K	1.0M
tha	0.7M	39.1K	23.1K	39.7M
tur	4.1M	40.2K	19.9K	483.0M
uig	5.2M	8.6K	4.3K	33.8M
urd	1.1M	46.7K	23.2K	14.4M
uzb	4.2M	42.5K	21.7K	60.3M
yor	2.1M	47.9K	24.5K	7.0M

Table 1: Data split and size of monolingual data (tokens) for different languages. For parallel data, size refers to the number of English plus foreign language tokens.

[1] https://code.google.com/archive/p/word2vec/

[2] https://www.isi.edu/ ulf/uroman.html

[3] LDC2015E13, LDC2015E14, LDC2015E83, LDC2015E84, LDC2016E57, and LDC2016E86 to LDC2016E105

[4] LDC2012E51, LDC2012E24, (Pilevar et al., 2011)

4 Experiments

4.1 Machine Translation

We perform end-to-end machine translation experiments on 15 different languages described in Section 3. We use Giza++ (Och and Ney, 2003) to get the alignments and Moses (Koehn et al., 2007) to train and decode phrase based machine translation (PBMT) systems. The parallel data is stemmed to the first 4 characters for training the alignments but not for the PBMT system. We use 5 iterations of each of IBM models 1, HMM and 4 to train the alignments both before and after adding the lexicons. In order to reduce the effect of randomness, we tune and test each system three times and report the average scores. Our baseline is the system before adding the lexicons. We test both the effect of only adding the lexicon extracted from $p(f|e)$ translation table using the English word vectors (L_e), and adding both the lexicons ($L_e + L_f$). Table 2 shows the BLEU scores of running different experiments. The languages are sorted by the size of their monolingual data. The first five languages have less than $10M$ tokens of monolingual data.

	baseline	L_e	$L_e + L_f$	improve
tam	19.2	**19.3**	19.2	0.1
ben	8.1	**8.2**	8.0	0.1
hau	19.4	**19.6**	**19.9**	0.2
amh	11.5	**11.9**	11.2	0.4
yor	14.2	**14.6**	14.3	0.4
som	18.7	**19.1**	**18.9**	0.2
urd	15.6	15.2	**16.2**	0.6
spa	**40.0**	**40.0**	**40.0**	0.0
cmn	12.5	**12.7**	**12.7**	0.2
uig	12.8	**14.3**	**14.0**	1.2
tha	20.3	20.1	**20.5**	0.2
uzb	13.2	13.5	**13.9**	0.7
ara	**18.2**	18.1	18.0	-0.2
fas	19.2	19.3	**19.4**	0.2
tur	14.7	**15.4**	**15.4**	0.7

Table 2: Machine translation experiments (BLEU). For languages with less than $10M$ monolingual tokens (first five) we only use L_e, otherwise we use both lexicons L_e+L_f. This way we improve baseline for almost all languages.

We see that it is generally better to only use L_e for languages with small monolingual data and use both L_e and L_f for others. If we put the threshold at $10M$ tokens of monolingual data, we improve the BLEU score over baseline for almost all languages, up to 1.2 points for Uighur. The exceptions are Arabic and Spanish. However, the Spanish experiment is hardly within the low resource settings as it has about $24M$ tokens of parallel data.

4.2 Alignments

In addition, we perform word alignment experiments on Arabic/English, Chinese/English, and Farsi/English, for which we have access to gold alignment data (Section 3). We append the test sentences to the existing parallel training data for each language (Table 1) and use it to get the alignments. Baseline and proposed methods are defined as in the machine translation experiments above (Section 4.1). Note that word vectors for these three languages are trained on more than 10M tokens, so we use both lexicons in the proposed method. Table 3 presents the precision, recall, and f-score of the alignments compared to the gold alignments.

	baseline	$L_e + L_f$
ara	63.1/58.1/60.5	63.8/58.4/61.0
cmn	66.5/61.6/63.9	66.7/61.6/64.1
fas	52.7/66.7/58.9	54.3/68.5/60.6

Table 3: Word alignment experiments (alignment precision/recall/f-score). The proposed method ($L_e + L_f$) improves baseline in all cases.

The proposed method gets better precision, recall, and f-score for all three languages.

5 Conclusion

In this paper we present a method for improving word alignments using word similarities. The method is simple and yet efficient. We use off-the-shelf distributed word representation tools to encourage a subset of translation table entries that are common between semantically similar words. End-to-end experiments on translating 15 languages into English, as well as alignment-accuracy experiments for three languages, show consistent improvement over the baseline.

Acknowledgments

This work was supported by DARPA contract HR0011-15-C-0115. The authors would like to thank Ulf Hermjakob, Jonathan May, and Michael Pust for their comments and suggestions.

References

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(2).

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005* .

Colin Cherry and Dekang Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *Proc. COLING*.

Adrià De Gispert, Deepa Gupta, Maja Popović, Patrik Lambert, Jose B Marino, Marcello Federico, Hermann Ney, and Rafael Banchs. 2006. Improving statistical word alignments with morpho-syntactic transformations. In *Advances in Natural Language Processing*.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improve word alignment precision for syntax-based machine translation. In *Proc. Workshop on Statistical Machine Translation*.

Ulf Hermjakob. 2009. Improved word alignment with statistics and linguistic heuristics. In *Proc. EMNLP*.

Tomáš Kočiskỳ, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947* .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL, interactive poster and demonstration sessions*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural MT. In *Proc. Workshop on Neural Machine Translation*.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proc. NAACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proc. NAACL*.

Jeff Ma, Spyros Matsoukas, and Richard Schwartz. 2011. Improving low-resource statistical machine translation with a novel semantic word clustering algorithm. *Proc. MT Summit XIII* .

Coşkun Mermer and Murat Saraçlar. 2011. Bayesian word alignment for statistical machine translation. In *Proc. ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Robert C Moore. 2004. Improving IBM word-alignment model 1. In *Proc. ACL*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP*.

Mohammad Taher Pilevar, Heshaam Faili, and Abdol Hamid Pilevar. 2011. Tep: Tehran English-Persian parallel corpus. In *Proc. CICLing*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proc. EMNLP*.

Theerawat Songyot and David Chiang. 2014. Improving word alignment using word similarity. In *Proc. EMNLP*.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proc. EACL*.

Kristina Toutanova, H Tolga Ilhan, and Christopher D Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proc. EMNLP*.

Vuong Van Bui and Cuong Anh Le. 2016. Smoothing parameter estimation framework for IBM word alignment models. *arXiv preprint arXiv:1601.03650* .

Ashish Vaswani, Liang Huang, and David Chiang. 2012. Smaller alignment models for better translations: unsupervised word alignment with the l0-norm. In *Proc. ACL*.

Hui Zhang and David Chiang. 2014. Kneser-Ney smoothing on expected counts. In *Proc. ACL*.