Phrase-based Unsupervised Machine Translation with Compositional Phrase Embeddings

Maksym Del **Andre Tättar** **Mark Fishel**
Institute of Computer Science
University of Tartu, Estonia
{maksym.del,andre.tattar,fishel}@ut.ee

Abstract

This paper describes the University of Tartu's submission to the unsupervised machine translation track of WMT18 news translation shared task. We build several baseline translation systems for both directions of the English-Estonian language pair using monolingual data only; the systems belong to the phrase-based unsupervised machine translation paradigm where we experimented with phrase lengths of up to 3. As a main contribution, we performed a set of standalone experiments with compositional phrase embeddings as a substitute for phrases as individual vocabulary entries. Results show that reasonable n-gram vectors can be obtained by simply summing up individual word vectors which retains or improves the performance of phrase-based unsupervised machine tranlation systems while avoiding limitations of atomic phrase vectors.

1 Introduction

Most successful approaches to machine translation (Wu et al., 2016; Bahdanau et al., 2014; Vaswani et al., 2018; Gehring et al., 2017) rely on the availability of parallel corpora. Supervised neural machine translation (NMT) employs the encoder-decoder architecture, where the encoder reads the source sentence and produces its representation which is then fed to the decoder that tries to generate the target sentence word by word. Cross-entropy loss is usually used as a training objective and beam search algorithm is used for inference. These neural models show state-of-the art performance but rely on vast amounts of parallel data.

On the other hand, there is a Statistical Machine Translation paradigm that is based on phrase tables that are learned from parallel corpus. These methods are currently replaced by neural counterparts for high-resource languages, but perform better in low-resource settings (Bentivogli et al., 2016).

For some language pairs the size of parallel corpus ranges from extremely low to almost zero. These extremely low-resource language pairs were a motivation for the Unsupervised Machine Translation (Lample et al., 2017; Artetxe et al., 2017b) that aims to translate language without usage of the parallel corpora for training.

The first step of the unsupervised approach to translation is the same for all methods: learning word level embedding spaces for source and target languages and then aligning these spaces. Next, one of the unsupervised vector space mapping methods (Artetxe et al., 2017a; Conneau et al., 2017) is applied to align spaces together and perform word by word translation. This mapping is only possible because of the linear properties of the word embedding method that is used to get word vector representations. Lastly, to improve system's performance, iterative refining using either neural network models or parts of SMT pipeline is done (Mikolov et al., 2013b; Lample et al., 2018). In this work we implement a system that is similar to the latter approach.

Statistical Machine Translation systems that are based on the phrase representations require smaller amounts of data to achieve reasonable performance. Phrase-based Unsupervised Machine Translation is motivated by the assumption that methods working without parallel data can benefit from the usage of phrases as the basic units. In order for phrases to be used for Unsupervised Translation they have to be represented in a suitable way. Current approach is to learn phrase embeddings as atomic vocabulary units (Lample et al., 2018).

In this work, we implement systems for the English-Estonian language pair following the guidlines presented in (Lample et al., 2018) and

Proceedings of the Third Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, pages 361–367
Belgium, Brussels, October 31 - Novermber 1, 2018. ©2018 Association for Computational Linguistics
https://doi.org/10.18653/v1/W18-64034

show that simply using sum of individual word embeddings can produce reasonable phrase embeddings. This eliminates the need of having huge n-gram vocabulary that might be hard to learn and use in unsupervised bilingual mapping. It also allows to obtain embeddings for arbitrary phrases as opposite to using predefined limited set of phrases.

This paper is organized as follows: In Section 2 we describe our unsupervised translation system baseline. Section 3 describes our approach to computing phrase embeddings for arbitrary phrases for UMT. In Section 4 we describe our experiments for compositional phrase embeddings standalone and in context of UMT. Section 5 concludes the work.

2 Baseline UMT System

Our baseline systems relies on the recently proposed Phrase-based UMT framework (Lample et al., 2018). Firstly, we learn monolingual embeddings for words and then use unsupervised mapping for bilingual lexicon extraction. The lexicon is used to compute semantic distances between phrases which results in the phrase table as used in Statistical Machine Translation. Lastly, we use standard SMT pipeline [1] with ngram language models in order to generate translations. We do not perform iterative back translation for simplicity and due to time and compute limits.

In the simplest case the phrase table consists solely of the unigram entries, but following Lample et al. we also consider bigram and trigram expermients. However, we use a different procedure for making decisions on which ngrams to consider for adding to the phrase table.

Extraction of n-grams is done by probabilistically joining words into phrases. We use frequency filtering with sampling, so n-grams are sometimes joined and sometimes not. We downsample the most frequent words, with probability function $p = \frac{1}{f^{\beta}}$, where p is the sampling probability, f is the n-gram frequency and B is a small weight (we used $\beta = \frac{1}{8}$). Frequency filtering is used for very rare words, so they must appear more than the set threshold. For example a n-gram that appears 25 times is joined with probability 0.668, but n-gram that appears 1000 times is joined with probability 0.422.

The point of using phrases is that they should be less ambiguous than words, and the fact that one word in one language can be a phrase in another language, like Estonian word ”laualt”, which means ”from the table”. The extraction of n-grams is done as in Blue2vec algorithm (Tättar and Fishel, 2017). To compute embeddings (S. Harris, 1954; Mikolov et al., 2013b) we use FastText[2] (Bojanowski et al., 2017) embeddings instead of word2vec (Mikolov et al., 2013a). We prefer FastText because it produces embeddings that incorporate subword level information which is proven to be helpful.

After finding vectors for words and phrases, we need to project the source and target language embeddings into the same space (Artetxe et al., 2017a; Conneau et al., 2017; Artetxe et al., 2018). Projecting is done using the MUSE [3] library. We use cross lingual similarity scores to score N-grams.

3 Approach to Phrase Embeddings

State of the art Phrase Based UMT systems (Lample et al.) learn phrase embeddings as individual vocabulary entries (to form the entry, we just concatenate words together with underscore). Although this approach provides good embeddings for phrases, it has serious limitations. Firstly, it is very memory intensive because it is infeasible to learn and store vocabulary of all phrases that can occur in the corpus. The size of the vocabulary grows almost exponentially over the length of the phrase, and thus vocabularies of that order of magnitude do not fit into the computer memory in most cases. Secondly, there is a data sparsity problem since some (even two-word) phrases occur rarely even in the very large corpus (it makes learning embedding vectors hard for some phrases).

The idea to combine embeddings to get a phrase embedding (Mikolov et al., 2013b) was successfully used in context of tasks such phrase similarity (Muraoka et al., 2014) and non-compositional phrase detection (Yazdani et al., 2015). We follow similar idea and show why it is highly suitable specifically for unsupervised translation.

In summary, we first learn vectors for reasonable amount of phrases as single-token vocabulary entries (e.g. ”research_paper”). At the same time (as a part of the same training procedure), we learn embeddings for individual words (”research” and ”paper” separately). Finally, we train a regression

[1] http://www.statmt.org/moses/

[2] https://github.com/facebookresearch/fastText

[3] https://github.com/facebookresearch/MUSE

model to predict phrase vectors from their word vectors. We assume that the model will learn the function that captures the pattern of combining word vectors in order to generate phrase vectors. Since we can obtain vectors for arbitrary words, we thus can estimate vectors for arbitrary phrases by combining word vectors with the learned function. Remaining text of the subsection defines the pipeline in step by step fashion.

Step 1: obtain training data for non-compositional modeling. In order to compute vectors for words and subset of phrases (we treat phrases as single-unit vocabulary entries at this point) we first need to get a monolingual corpus and do some preprocessing like lowercasing / truecasing and tokenization.

Step 2: extract phrase candidates. Since we can not learn vectors for all phrases in the corpus, we have to decide which phrases to use to learn vectors for. One option is to randomly glue desired number of phrases together while other options include only gluing phrases that belong to some specific set of desired phrases. The set can be formed by scoring all phrases from the corpus by some criterion and then taking top N phrases based on their scores. Here we provide non-comprehensive list of criterion that can serve to the purpose of gluing two-word phrases: Likelihood ratio, Raw frequency, Poisson Stirling criterion, Chi square score, Dice score, Jaccard measure etc. We refer author to external literature for additional information on this topic (e.g. (Manning et al., 2008)). Concrete metric should be task specific or empirically chosen.

Step 3: glue phrases. At this step we simply go through the corpus from Step 1 and glue some words together based on the phrases set from the Step 2.

Step 4: train word embedding model. At this step we train (say) the Skip-gram model on words and phrases corpus from Step 3. This way we get semantic vectors for words and some phrases.

Step 4: obtain training data for compositional modeling. At this step we extract phrase vectors for phrases that we glued at the Step 3. Then we extract word vectors for words that are used to compose these phrases. The dataset then consists of following pairs of entities: sequence of the word vectors as an input, and phrase vector as the target.

Step 5: train compositional model on the data from the Step 4. At this point we use the dataset from previous step to teach the model to compose word vectors in a way that phrase vector is produced

One critical property of this framework is that it produces vectors for phrases as if they were learned as the single-token units as a part of the vocabulary. That is, result phrase embeddings will not differ (in terms of their properties) from word embeddings. Word embeddings are learned as individual vocabulary units and satisfy to all assumptions of bilingual mapping methods (Artetxe et al., 2018); therefore, phrase embeddings learned this way would also do. Since billingual embedding mapping is the key necessury step of all current approaches to UMT, our phrase embeddings might become strong alternative for any existing UMT system.

We make our implementation of the pipeline available as an open source project [4].

4 Experiments

4.1 Compositional Phrase Embeddings

In this subsection we describe our experiments on compositional phrase embeddings standalone.

4.1.1 Setup

We explored on the predictive ability of the different variants of the compositional models that we train as a part of the Step 5 of the framework. Following steps describe concrete decisions we made as a part of our implementation of the general pipeline we defined in previous subsection.

Step 1: obtain training data for non-compositional modeling. We used first 1 billion bytes of English Wikipedia as our training data. The data contains 124,301,826 lowercased tokens.

Step 2: extract phrase candidates. We only glued phrases that belong to some specific set of desired phrases. The set was formed by scoring all phrases from the corpus by likelihood ratio criterion and then taking top 600,000 phrases based on their scores.

Step 3: glue phrases. At this step we simply went through the corpus from Step 1 and glued some words together based on the phrases set from the Step 2.

Step 4: apply skip-gram model. At this step we trained Skip-gram model on words and phrases

[4] https://github.com/maxdel/bigram_embedder

corpus from Step 3. This way we got semantic real valued vectors for words and some phrases. We trained the system using *fasttext* framework ((Bojanowski et al., 2017)) for 6 epochs with default parameters expect for the embedding size which we set to 100.

Step 4: obtain training data for compositional modeling. The result dataset size was about 600,000 training examples.

Step 5: train compositional model on the data from the Step 4. At this point we used the dataset from previous step to teach different models to compose word vectors in a way that phrase vector is produced.

We also left some examples apart for validation and testing. The test set size was 2400 examples, development test size was 2000 examples. Development set was used to tune models hyperparameters, and test set was used to perform final models comparison.

4.1.2 Candidate Models

Let *w1* be the vector of the first word and *w2* the vector of the second. Let also *p* be the result vector of the phrase and D be the dimension of the w1, w2, and p. The types of the models that we implemented and trained[5] are following.

- Simple addition *(AddSimple)*:

$$p = w1 + w2$$

- Addition with attention weights *(AddAtt)*:

$$p = a1 * w1 + a2 * w2$$

  where a1 and a2 are scalars that are learned by first concatenating the word vectors, and then projecting result into two dimensions.

- Dimwise addition with attention weights *(AddAttDimwise)*:

$$p = a1 * w1 + a2 * w2$$

  where a1 and a2 are vectors of the size D that are learned by first concatenating the word vectors, and then projecting result into the D dimensions. Therefore, we add two vectors with weight assigned to each dimension.

- Neural Network with one linear layer *(Linear)*:

$$W1 * ([w1, w2])$$

  where W1 is parameters matrix and [w1, w2] means concatenation

- Neural Network with dense ReLU layer and linear layer *(NonLinear)*:

$$W2 * ReLU(W1 * [w1, w2])$$

- Multilayer Neural Network *(MultilNonLinear)*: the same as the previous one, but with two more nonlinear layers. The sizes of hidden layers are 170, 130, and 100.

- Long Short Term Memory network *(LSTM)*: last timestep is used as phrase representation ((Hochreiter and Schmidhuber, 1997)).

Smooth l1 loss was used in order to train all the models:

$$\text{loss}(x, y) = \frac{1}{n}\sum_i z_i,$$

where

$$z_i = \left\{ \begin{array}{ll} 0.5(x_i - y_i)^2, & \text{if}|x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & \text{otherwise} \end{array} \right\}$$

We choose this loss because it is more tolerant to outliers, which may occur due to non-compositionality of some phrases.

4.1.3 Experiments Results and Analysis

In order to get interpretable accuracy scores for models comparison we first run our trained regression models in the inference mode to predict phrase vectors for the test set. Then we retrieve the top N (N is one of {1,3,5,10}) closest points in the embedding space, and check if the ground truth phrase belongs to this set. If the phrase is not in the topN, we count it as an error. Lastly we divide number of non-error examples by the size of the training set to obtain accuracy score. Table 1 shows accuracy scores across various models across various topN values.

As we can see from the Table 1, simple summation baseline shows decent performance in compositional phrase vector modeling. This interesting result was explained by the authors of the Skip-gram model ((Mikolov et al., 2013b)). They

[5] *AddSimple* model does not require training while attention weights for *AddAtt* and *AddAttDimwise* are learned from data

Table 1: Accuracy results of explicit evaluation of compositional models. Top 3 results among columns are in bold.

Model	**top1**	**top3**	**top5**	**top10**
AddSimple	***0.35***	**0.81**	**0.88**	**0.94**
AddAtt	0.37	0.65	0.74	0.84
AddAttDimwise	0.38	0.66	0.75	0.84
Linear	**0.71**	**0.85**	**0.88**	**0.92**
NonLinear	0.62	0.75	0.80	0.85
MultiNonLinear	**0.69**	0.83	0.87	0.91
LSTM	**0.73**	**0.88**	**0.92**	**0.95**

show that addition of two token vectors approximately equivalent to the AND operation between their distributions over context words (we predict context / surrounding words with the Skip-gram model). This means that the result token vector will be equivalent to the token (phrase or word) that shares the same context with the input token vectors. Despite the good performance of the simple addition function, we observe drop in performance for attentional analogues. This might be due to the fact that it is sometimes hard to predict these attention weights from the words themselves, since the Skip-gram embeddings does not really contain much of Part of Speech (POS) information (e.g. words like ”go”, ”goes”, ”going”, ”went” are grouped together despite having different POS tags) while this information is what was needed to achieve good results in (Muraoka et al., 2014). Among neural architectures, the LSTM network was able to outperform simple sum operation. It might be due to the separate gates it uses for memorizing the important (in context of the future phrase) semantic part of the word, forgetting redundant dimensions, and updating the first word with some information from the second word. We conducted experiments on phrases with length up to two words for simplicity.

This experiments show that LSTM network is a powerful tool for predicting compositional phrases while linear layer remains a strong option. However, we also strongly consider summation as a valid option due to its simplicity, comparable performance, and theoretical motivation. In fact, we use summation powered phrase embeddings in our ongoing experiments with phrase-based UMT system described here.

Note that sum shows low performance at top1 sampling (Table 1). It is explainable since there is no information about the order in which individual words were summed so model just outputs the vector for a phrase with reverse word order. However, it is still acceptable vector as it is shown by a huge jump at top3, where usually both word ordering options are included.

4.2 Unsupervised Machine Translation

In this subsection we describe our experiments with compositional phrase embeddings as a part of the phrase based UMT system.

4.2.1 Setup

In this subsection, we describe the the setupparameters we used in our system. The UnsupervisedMT framework [6] was used to train baselines.

Data. In our experiments we use datasets from the WMT’18 unsupervised translation task (Bojar et al., 2018) for Engish-Estonian language pair. 15M monolingual sentences for Estonian and the same amount for English is considered.

Preprocessing. We use tokenization and truecasing to preprocess our data for both embeddings learning and language models training.

Unigram system. This system contains unigram cross-lingual word embeddings and does not include the n-grams.

Bigram-atomic/trigram-atomic system. This systems use phrase table that consist of bigram/trigram entries. Embedding vectors for bigrams/trigrams were obtained as a result of treating ngrams as atomic vocabulary units.

Bigram/trigram-sum. This system uses phrase table that consist also of bigram/trigram entries, embedding vectors for which were obtained as a result of summing individual word vectors that form the bigram. The proportion of unigram / bigram / trigram types is about 0.5 / 0.25 / 0.25 for English and 0.89 / 0.9 / 0.2 for Estonian.

N-gram extraction. Frequency filter settings are the following: 20, 120 and 90 (frequency counts for filtering out infrequent n-grams, unigrams, bigrams and trigrams respectively). The beta parameter is set to 0.125. This procedure is used to extract ngrams for both the atomic and sum systems. As a result, for bigram experiments, 38% of English vocabulary and 9% of estonian vocabularty consisted of bigrams.

FastText Embeddings. We use CBOW with character n-grams of size 3 to 6 as a core algo-

[6] https://github.com/facebookresearch/UnsupervisedMT

rithm for embeddings learning. The number of dimensions is set to 300. Other parameters are kept as default.

Bilingual MUSE embeddings. Default parameters are kept and MUSE is used on CUDA GPUs. Embeddings dimensions are set to 300.

Language model. We train Moses style ngram langage models with the order of 5.

Moses and beam search. We keep all Moses and beam search hyperparameters default.

4.2.2 Experiments Results and Analysis

Final BLEU scores for the experiments with Unigram, Bigram/Trigram-atomic, and Bigram/Trigram-sum systems are presented in Table 2.

Table 2: BLEU scores for our baselines and systems powered with compositional phrase embeddings for Estonian-English and English-Estonian language directions.

System	**et-en**	**en-et**
Unigram	4.42	4.14
Bigram-atomic	7.92	4.73
Trigram-atomic	7.63	4.96
Bigram-sum	6.25	3.88
Trigram-sum	6.28	4.05

Regarding Ngram-atomic series of experiments, Table 2 shows that there is an advantage of using bigrams for both language directions. However, for Estonian-English the advantage is much bigger (+3.5 against +0.59 BLEU). That might be due to the fact the the number of ngrams in English vocabulary is more then 3 times as big as the corresponding number for Estonian vocabulary. Trigram experiment provides no significant increase or decrease over the bigram experiment.

The benefit of using bigrams and the trigram trend are consistent with the findings of Lample et al.. However, while Lample et al. reports the increase of about 1 BLEU point when using bigrams, we observe the increase of 3.5 for et-en. Note that we also use custom ngram extraction procedure as opposite to taking top N most frequent bigrams (Lample et al., 2018).

In case of Ngram-sum experiment for et-en, the system outperforms unigram experiment suggesting that the compositional embeddings do have semantic power. However, it is below the Bigram-atomic and Trigram-atomic baselines which is expected since the predictivness of the compositional model is not perfect. For en-et however, neither Bigram-sum nor Trigram-sum system outperforms atomic baseline suggesting that the topic needs additional dedicated research efforts.

5 Conclusions and Future Work

In this work, we present our results for the WMT18 shared task on unsupervised translation. Our baseline systems follow principles of the Phrase-based Unsupervised MT where we study unigram, bigram, and trigram systems. The vectors for ngrams are learned as individual vocabulary entries which has its limitations. Thus we study compositional phrase embeddings as a substitute, and show that simply summing up individual phrase words results in phrase embeddings that allow UMT systems to improve over baselines.

We showed that atomic phrase embeddings can be accurately estimated with compostional predictive models. Still, the effect of compositional phrase embeddings on PBUMT is still to be studied. More language pairs should be considered and more exhaustive targeted experiments with stronger baselines should be done. We leave this research direction for future work.

Acknowledgments

This work was supported by the Estonian Research Council grant no. 1226.

The authors would like to thank the University of Tartu's Institute of Electronics and Computer Science for providing GPU computing resources.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017a. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5012–5019.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017b. Unsupervised neural machine translation. *CoRR*, abs/1710.11041.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. *CoRR*, abs/1608.04631.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of WMT'18: the Third Conference on Machine Translation*, Brussels, Belgium.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *CoRR*, abs/1711.00043.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *CoRR*, abs/1804.07755.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. 2014. Finding the best model among representative compositional models. In *PACLIC*.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Andre Tättar and Mark Fishel. 2017. bleu2vec: the Painfully Familiar Metric on Continuous Vector Space Steroids. pages 619–622.

A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, Ł. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. 2018. Tensor2Tensor for Neural Machine Translation. *ArXiv e-prints*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Majid Yazdani, Meghdad Farahmand, and James Henderson. 2015. Learning semantic composition to detect non-compositionality of multiword expressions. In *EMNLP*.