The Prague Bulletin of Mathematical Linguistics
NUMBER 108 JUNE 2017 271–282

Providing Morphological Information for SMT Using Neural Networks

Peyman Passban, Qun Liu, Andy Way

ADAPT Centre, School of Computing, Dublin City University, Ireland.

Abstract

Treating morphologically complex words (MCWs) as atomic units in translation would not yield a desirable result. Such words are complicated constituents with meaningful subunits. A complex word in a morphologically rich language (MRL) could be associated with a number of words or even a full sentence in a simpler language, which means the surface form of complex words should be accompanied with auxiliary morphological information in order to provide a precise translation and a better alignment. In this paper we follow this idea and propose two different methods to convey such information for statistical machine translation (SMT) models. In the first model we enrich factored SMT engines by introducing a new morphological factor which relies on subword-aware word embeddings. In the second model we focus on the language-modeling component. We explore a subword-level neural language model (NLM) to capture sequence-, word- and subword-level dependencies. Our NLM is able to approximate better scores for conditional word probabilities, so the decoder generates more fluent translations. We studied two languages Farsi and German in our experiments and observed significant improvements for both of them.

1. Introduction

Phrase-based SMT (PBSMT) (Koehn et al., 2003) is the state-of-the-art model for providing automatic translations, but it suffers from serious problems. The performance of the PBSMT model considerably decreases in the presence of large vocabularies and a high rate of out-of-vocabulary words. These phenomena are closely tied to morphology-related issues frequently encountered in MRLs. Recently, neural machine translation (NMT) (Cho et al., 2014) has appeared as a very powerful alternative

 Corresponding author: peyman.passban@adaptcentre.ie
Cite as: Peyman Passban, Qun Liu, Andy Way. Providing Morphological Information for SMT Using Neural Networks. The Prague Bulletin of Mathematical Linguistics No. 108, 2017, pp. 271–282.
doi: 10.1515/pralin-2017-0026.

for PBSMT, which is able to generate competitive or in some cases even better results. However NMT suffers from the same problems. Incorrect word selection and generating wrong surface forms are direct consequences of such shortcomings. Accordingly, both paradigms have problems with MRLs, some of which we try to address here. Although we benefit from neural-network-based features, the main interest of the paper is the SMT approach and its enhancement, so we do not study NMT engines.

SMT can be viewed as a sequential pipeline which takes a sentence in a source language, manipulates it step by step and finally produces a target sentence. In such a multi-step process the most compatible data distribution is selected to train the best (task-specific) model. Data selection techniques are designed in this regard. Afterwards the training data is preprocessed during normalization and tokenization to be more readable/understandable for other subsequent steps. Source and target words are aligned to find cross-lingual lexical mappings. Phrases are extracted and models are trained correspondingly. At the final stage the best counterparts of source phrases are discovered through a search-based solution. Target phrases are combined together to make a coherent and fluent translation. In special cases some post-translation processing is also applied to the final translation. All of these steps are carried out using statistical models which rely heavily on word co-occurrences.

Neural models are known as powerful techniques to capture semantic information. They provide richer information than count-based and statistical models. There are several research papers which boost the aforementioned SMT sub-modules via neural techniques. Duh et al. (2013) uses neural networks (NNs) to select better sentences to train high-quality SMT engines. Tamura et al. (2014) explore neural alternatives instead of the EM-based model for word alignment. Li et al. (2014) design a neural reordering function.

In this paper we also try to model morphological information using NNs. To this end we propose two solutions: (i) we introduce morphological features for the factored translation model (Koehn and Hoang, 2007), and (ii) we manipulate a language model (LM) to incorporate morphological information.

The factored translation model (FTM) is one of the most suitable frameworks to include different annotations at decoding time, such as morphological information. The main problem with PBSMT is that it translates text phrases without any explicit use of linguistic information, which seems crucial for a fluent translation. In FTMs each word is extended by a set of annotations, so that a word in this framework is not only a token but a vector of factors, e.g. a simple word in PBSMT can be represented by a vector of {*word (surface form), lemma, part-of-speech (POS) tag, word class, morphological information*} in its factored counterpart. Clearly the new representation is richer than the word's surface-form. Since the main focus in FTMs is on word-level enrichments, it addresses the problem of morphology which is the main interest of this paper.

In word- or phrase-based approaches, each word is treated independently, i.e. *'studies'* has no relation to *'studied'*. If only one of them was seen during training, translation of the other one would be hard (or impossible) for any SMT engine, even

though both words come from the same stem. Translation knowledge of their shared stem along with auxiliary morphological information could help us translate both of them. This property not only provides solutions for these types of morphological issues but also addresses the data sparsity problem at the same time. A factored translation model follows a similar approach and performs better than other models (which rely on surface forms) for MRLs.

Translation in FTMs is generally broken up into two translation and one generation steps. A source lemma is translated into a target lemma. Morphological and POS factors are translated into target forms and the final form is generated based on the lemma and other factors. Factored models follow the same implementation framework as the phrase-based model. In these models the translation step operates at the phrase level whereas generation steps are word-level operators. For more information on FTMs see Koehn and Hoang (2007). In our modified FTM we have four factors of the *surface form*, *lemma*, *POS tag* and *morphology tag* for each word. It is clear how the first three factors are defined. The last factor is based on morphology-aware embeddings. First we train word embeddings (see Section 2.1) which preserve subword-level and morphological information. Then we cluster words based on their embeddings. The cluster label of each word indicates its morphological tag.

As previously mentioned, along with the translation model we try to enrich an LM. The LM is the main source of monolingual knowledge in translation which plays a key role in providing fluent translations. This module is the best means by which we can directly impose morphological constraints. In PBSMT models n-gram LMs are explored, whereas we benefit from a neural variant in our case. We selected Farsi (Fa) and German (De) for our experiments. Farsi is a morphologically rich and a low-resource language. Therefore, any small improvement in such a language could be a valuable achievement. Beside Farsi experiments we also evaluate our models on German. This language is well-studied in the field of MT and there exist plenty of experimental studies on German, but we use it to provide better comparisons with previous work and show the strength and weakness of our models.

2. Proposed Models

2.1. Enriching Word Embeddings with Subword Information

Words are not always usable in their original forms, as they are symbolic units and need to be transformed into numerical forms for some applications. Each word carries a particular type of information and has specific syntactic and semantic roles. The word's relation with other constituents is also a key property which is defined exclusively for each word. Considering all of these features, it is quite challenging to find a numerical counterpart for a word, which preserves all of these properties and represents the same word in a numerical feature space. To this end there are well-known models such as Salton et al. (1975) which try to transfer words and their syntactic

and semantic information. Recently, NNs have become the established state-of-the-art for creating distributed representations of words (and also other textual units such as characters etc.). Hinton (1986) proposed an NN-based embedding model for the first time, introducing the idea of a *shared learning space*, where the embeddings (word vectors) themselves are also trainable parameters of the model.

Word embeddings are real-valued representations in an n-dimensional feature space. Recent work has shown that these distributed representations can preserve meanings, as well as semantic and syntactic dependencies. However, existing word-based models have some deficiencies, especially with regard to MRLs. In these models, each word is treated as an atomic unit which is not an appropriate way of processing MCWs. In this section we propose a new technique designed to model intra-word relations. Word-based models (Mikolov et al., 2013; Pennington et al., 2014) are not able to (efficiently) transfer rare words. Our model composes word embeddings from subunit embeddings and tries to solve this problem.

There are several models to train word embeddings. One of the most successful models is *Word2Vec* proposed by Mikolov et al. (2013). Almost all other work has followed this unsupervised approach. The main intuition behind our model is the same, but the internal operation is quite different. *Word2Vec* is a simple feed-forward model in which a target random word of an input sequence is selected to be predicted by means of its surrounding context. Word vectors are updated with respect to error values of the prediction phase. More formally the network tries to compute $P(w_i|C)$ where w_i is the target word and C indicates its context. In the simplest scenario the context C is the preceding word just before the target word and the network includes one hidden layer h with the weight matrices $W_{i:h} \in \mathbb{R}^{|input| \times d}$ and $W_{h:o} \in \mathbb{R}^{d \times |\mathcal{V}|}$. $\mathcal{V}$ is the vocabulary set and d is the size of h. The probability of each word given its context is estimated via a *softmax* function, which is a scalar that maps values of its input vector into the range $[0, 1]$, so that new values can be interpreted as probabilities. This scalar is formulated as in (1):

$$P(w_i = j|C) = \frac{\exp(h_t.w^j + b^j)}{\sum_{j' \in \mathcal{V}} \exp(h_t.w^{j'} + b^{j'})} \tag{1}$$

where w_i is the j-th column of $W_{h:o}$ and b_j is a bias term. Input to the *softmax* function is $h_t \in \mathbb{R}^d$ and its output is $v \in \mathbb{R}^{|\mathcal{V}|}$. The j-th cell of v is interpreted as the probability of selecting the j-th word from $\mathcal{V}$ as the target word. Based on *softmax* values the word with the highest probability is selected and the error is computed correspondingly. Error values are back-propagated to the NN in order to update network parameters. Word embeddings are part of those parameters which are updated.

Our model is a simple extension of the basic *Word2Vec* model. In the basic model the surface form of words are taken into account, whereas we segment each word into its stem and affixes, and the embedding of the surface form of each word is a composition of its subunit embeddings, i.e. the embedding of w_i is obtained by $\mathcal{E}(w_i) = \left(\sum_{m \in \mathcal{M}(w_i)} \mathcal{E}(m)\right) + \mathcal{E}(w_i)$, where $\mathcal{E}$ is the embedding form. w_i may have

several morphemes (subunits) where $\mathcal{M}(w_i)$ is the set of all possible subunits of w_i. We show an example from our training corpus to clarify the mechanism of making word embeddings. For the given word *'pre.process.ing.s'* the embedding is generated with this computation: $\mathcal{E}(\mathrm{pre}) + \mathcal{E}(\mathrm{process}) + \mathcal{E}(\mathrm{ing}) + \mathcal{E}(\mathrm{s}) + \mathcal{E}(\mathrm{preprocessings})$.

In our model we treat the word's surface form as another internal subunit, because it makes the approach more robust to noisy morphological segmentations. This strategy also generates better embeddings. We process all sentences of our training corpora. Words are segmented using *Morfessor* (Smit et al., 2014). We have a unique embedding vector for each subunit in our neural architecture. Subunit embeddings reside in a look-up table whose values are updated during training. Based on the input sentence the target word is randomly selected and the context vector is generated. Each word's embedding in the context vector is a linear combination of its subunits. The model tries to predict the correct target word at the output layer. Based on the prediction the error value is computed and back-propagated to the network. In the back-propagation pass all network parameters including word and subunit embeddings are updated. After training the model we obtain high-quality embeddings which have information about morphological properties and subunits of words. In Section 3 we show the impact of using such embeddings in SMT engines.

2.2. Training Subword-Aware Neural Language Models

An LM measures how likely a sequence of words is to occur in a text. It addresses the fluency of the given sequence, so that a sequence with a good word order has a high probability. The leading types of LMs are count-based or n-gram models which function based on the Markov chain assumption. In such models the probability of a sequence is computed by the conditional probabilities of words given their history: $P(S) = P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i|w_1^{i-1})$, where S is the given sequence with the length m. The model conditions the probability of each word over a chain of preceding words. Since computing the probability over the entire chain is not computationally feasible, it is usually limited to a bounded set of n previous words: $P(w_i|w_1^{i-1}) \simeq P(w_i|w_{i-n}^{i-1})$. The assumption states that the probability of a word is affected by its n preceding words. Obviously, a long history is preferable but such an assumption is made because of computational restrictions and limited data resources. These models are known as n-gram models and the limited-history problem is the main disadvantage of these models.

Recently, NLMs have been proposed as better alternatives for conventional LMs. NLMs are able to compute the word conditional probabilities over the entire chain and mitigate the history problem. They benefit from recurrent neural networks (Zaremba et al., 2014). As the name of these networks shows they have a recurrent mechanism; they process the input sentence word by word. At each step one word is taken as an input and the hidden state(s) is updated correspondingly. This loop continues until visiting the end of the sequence. When the process ends a summary of the entire

sequence resides in hidden states. As the network has access to such rich information it is able to provide a better estimation of word probabilities.

Although NLMs mitigate the history problem, similar to embedding models they also have serious problems with MRLs. In order to make NLMs compatible with MRLs, different models work at morpheme and character levels. We also propose a new hybrid (morpheme+character-level) model. For our NLM we could fine-tune the same architecture as in Section 2.1 (linear combination of subword embeddings), but character-aware models outperform subword-based NLMs. The state-of-the-art model for neural language modeling is the model by Kim et al. (2016) which relies on characters. Therefore, we also prefer to build our NLM over character-aware models.

In the character-aware framework words are segmented into characters. Each character has a dedicated embedding. All character embeddings are combined through a convolutional module. There is a set of different filters with different widths. The idea behind using different filters is to capture different n-gram information where the size of n-gram corresponds to the filter width. The maximum value of each filter, which is the most representative feature is selected to be combined with other maximum values from other filters. The combination of maximum values makes up the word's surface-form embedding. Word embeddings are passed through a highway layer (Srivastava et al., 2015) to make richer information for the following modules. The output of the highway layer is consumed by a Long Short-Term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997). LSTMs are memory-augmented recurrent models. Simple recurrent models are not able to model long-distance dependencies, but through an internal memory unit defined for LSTMs, they are able to remember the relation among words much better than simple models.

Our model is a simple extension to the character-aware NLM. The main responsibility of the convolutional module is to find relations among characters by using different filters. Instead of this neural computation we define a simpler but more straightforward technique to capture the same type of information. There are sets of consecutive characters in training corpora which always appear together. Since these characters are tied to each other and appear together, we do not decompose them, which means that instead of finding the relation of such a set of characters via different filters, we keep them together and explicitly inform the model about their relation. Therefore, we do not change the neural architecture but rather define a new preprocessing method.

In our model we extract all possible character n-grams. Each word with the length l (characters) could have up to $\frac{l \times (l+1)}{2}$ character n-grams, e.g for the word *'the'* we can extract these character n-grams: {*'t'*, *'h'*, *'e'*, *'th'*, *'he'*, *'the'*}. For each word, first we separate n-grams which are frequent. We keep those blocks (sets of consecutive characters which make the n-gram) as they are and do not segment them. Then we decompose the reminder into characters (if they are not frequent). In this model we start from higher order n-grams, i.e. for a word with l characters we start from $(l-1)$-grams. If

we cannot find a frequent subunit in the higher order (such as $l-1$), we look at lower orders (such as $l-2, l-3, ..., 2$). When we find a frequent l'-gram in a word, this means that there was no frequent character n-gram where $n > l'$. By use of an example from our Farsi[1] corpus (see Section 3) we try to clarify our segmentation method. For the word *'prdrāmdtrynhā'* meaning *'the people with the highest salary'*, the first frequent substring extracted is *'āmd'* which is a 3-gram constituent. This means that there is no frequent n-gram with $n > 3$. *'āmd'* also has the highest frequency among all other 3-grams, so in the presence of several frequent n-grams we select the most frequent one. *'āmd'* is separated and the segmentation model is applied to its preceding and following substrings. Each substring is considered as a new input to the model. We recursively apply the same procedure until all frequent substrings have been separated, which are *'āmd'*, *'dr'* and *'hā'* for this example. There are still three substrings remaining, namely *'pr'*, *'tr'* and *'yn'*. These three substrings are not considered as frequent in our setting, so they are all decomposed into characters. The final decomposition result by the proposed model is: *'prdrāmdtrynhā'* $\Rightarrow$ *'p.r.dr.āmd.t.r.y.n.hā'*. In our experiments we consider a constituent as frequent if it occurs more than 100 times in the entire training corpus.

Our NLM is the same as that of Kim et al. (2016) with one main difference. In the input layer of our model we have blocks instead of characters. Each block could include one or many characters. By using the character blocks we keep related characters together which means we do not need a convolutional (or any other neural) procedure. We explicitly define such information for the network through our blocks, and the convolutional module is a complementary layer to provide richer information about the relation of characters. Using this simple technique we are able to boost the character-aware NLM. We can use the same mechanism as in Section 2.1 in our NLM, namely each word can be represented via a linear combination of its subunits. Botha and Blunsom (2014) implemented this idea for language modeling and considerably improved the performance of previous NLMs. Although this model was quite successful, the model of Kim et al. (2016) outperforms it. Accordingly, we built our NLM via the character-aware model. Table 1 illustrates a simple comparison of these three approaches and shows the impact of our model.

Model	**German (De)**	**Farsi (Fa)**
Botha and Blunsom (2014)	296	-
Kim et al. (2016)	239	128
Proposed Model	**225**	**110**

Table 1. Perplexity scores of different NLMs (lower is better).

[1]We use the DIN transliteration standard to show the Farsi alphabets.

The table reports perplexity scores for different NLMs. The numbers reported for the German experiments from the first two models are taken from Kim et al. (2016). The German models are trained and evaluated on the same dataset as reported in the original paper (Kim et al., 2016). For the Farsi model we selected a corpus of 1 million words from the TEP++ corpus (see Section 3). We selected 1 million words to have the same size with the German corpus. The source code for the character-aware model is publicly available and we can run it on our Farsi corpus. Therefore, as we do not have access to the original morpheme-aware model, it is not possible to report its perplexity over the Farsi corpus.

3. Experimental Study

In this section we evaluate our models on Farsi and German. We selected Farsi as it is a morphologically rich and low-resource language. Because of such difficulties it is quite challenging to develop a reliable MT model for this language. Accordingly, we propose such complementary techniques to enrich existing models. German is one of the well-studied languages in the field of MT and there are plenty of resources and models for this language. Generally, German models are high-quality models and their translation and language models are rich enough to provide acceptable results. Because of large datasets, German models are diverse and cover almost all cases (words, phrases etc.), so they do not need such complementary techniques. It is also hard to show the impact of auxiliary information (morphological information in our case) for German as small improvements are usually lost in the presence of large datasets. Nonetheless we report German results for comparative purposes with acceptable improvements.

In the first experiment we trained De↔En and Fa↔En SMT models. To train the engines we used the TEP++ (Passban et al., 2015) and WMT-15 datasets[2] for Farsi and German, respectively. TEP++ is a collection of ~600K parallel sentences. We used 3K sentences each for testing and tuning, and the rest of the corpus for training. From the En↔De dataset we randomly selected 2M sentences for training. The German model is evaluated on newstest-2015 and tuned using newstest-2013. Our models are trained using Moses (Koehn et al., 2007) with its default configuration. The evaluation metric is BLEU (Papineni et al., 2002) and language models are trained on the target side of our corpora with SRILM (Stolcke, 2002). Language models are 5-gram models. In our FTMs, English and German words are lemmatized via the NLTK toolkit (Bird, 2006) and tagged using the Stanford POS tagger (Toutanova et al., 2003). For Farsi, words are lemmatized with an in-house lemmatizer and tagged with our neural model (Passban et al., 2016b). The English tagger uses the Penn Treebank tagset with 36 tags. The German model uses the STTS[3] tagset with 54 tags and the Farsi

[2] http://www.statmt.org/wmt15/translation-task.html.

[3] http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html.

model has 37 tags. Table 2 shows the impact of incorporating our embeddings into the SMT pipeline.

Direction	Baseline	$\textbf{Extend}_3$	$\textbf{Extend}_4^w$	$\textbf{Extend}_4^m$
En→De	21.11	21.42	21.57	**21.70**
De→En	29.50	29.58	29.71	**29.78**
En→Fa	21.03	22.14	22.27	**22.61**
Fa→En	29.21	30.53	30.67	**30.91**

Table 2. Enriching the FTM using morpheme-aware word embeddings.

In Table 2 **Baseline** is a PBSMT model and $\textbf{Extend}_3$ is an FTM with 3 factors of {*word, lemma, POS tag*}. $\textbf{Extend}_4^w$ and $\textbf{Extend}_4^m$ show factored models with additional morphological factors (4-factor models), where the first one relies on surface-form word embeddings (*Word2Vec*) and the second on our morpheme-aware embeddings. Word embeddings by nature are real-valued vectors, so they can be easily clustered. The cluster label of a word conveys morphological, syntactic and semantic information about the word. In our training mechanism we highlighted morphological information, so the cluster label could be interpreted as the morphology tag of words which defines the fourth factor. Bold numbers indicate that improvements are statistically significant compared to **Baseline** according to paired bootstrap re-sampling (Koehn, 2004) with $p = 0.05$.

Since Farsi and German are more complicated languages compared to English, we assign 1000 clusters for them and English words are categorized into 200 clusters. As Table 2 shows, the 3-factor model ($\textbf{Extend}_3$) outperforms the baseline PBSMT model. This is an expected result because FTMs are better alternatives for MRLs. The performance obtained by the 3-factor model could be further enhanced via word embeddings. The fourth factor in $\textbf{Extend}_4^m$ provides morphological information which is useful for the decoder to cope with complicated morphological constituents. We have a comparison between basic surface-form and morpheme-aware embeddings. $\textbf{Extend}_4^w$ is based on *Word2Vec* embeddings which inform the decoder with some general and high-level information about words. Such information is useful but not as impactful as the information provided by $\textbf{Extend}_4^m$, which relies on morpheme-aware embeddings and thus provides more specific/relevant information. This comparison demonstrates that the mechanism used in training our embeddings is able to capture morphological information.

In addition to the first experiment we designed another experiment to show the impact of the subword-aware NLM. The baseline model in Table 3 is a PBSMT model with a 5-gram language model. There are several ways to embed an NLM into the SMT pipeline. We could use the NLM to re-rank translation results. We could also

Direction	Baseline	n-gram^{w}	n-gram^{m}	Direction	Baseline	n-gram^{w}	n-gram^{m}
En→De	21.11	21.53	**21.88**	En→Fa	21.03	21.86	**22.36**
De→En	29.50	29.87	**30.43**	Fa→En	29.21	29.91	**31.05**

Table 3. Re-scoring word n-grams with NLMs.

restructure the decoder to score translation hypotheses by the NLM. We chose a third way which re-scores the word n-grams of the existing non-neural LM, i.e. we manipulate the n-gram LM with the NLM. The n-gram LM includes word n-grams and their associated scores (scores which are computed based on the word co-occurrences and the Markov chain assumption). We recompute those scores with our NLM and substitute the new scores with previous ones. In this experiment we use an LM whose word n-grams come from the statistical 5-gram model and their associated scores are computed by the subword-aware NLM. In Table 3 our NLM-based model is shown with **n-gram**^{m}. Results show that decoding with new scores is quite effective and improves translation performance. Along with our subword-aware NLM we trained another NLM which is a two-layer LSTM model and works over words (surface forms). We repeated the language-modeling experiment and re-scored word probabilities with the word-based LSTM model. The final system is **n-gram**^{w}. Although the LSTM-based model enhances the baseline model, its impact in not as great as **n-gram**^{m}. This comparison confirms that morphological information provided by **n-gram**^{m} is more impactful than those of the word-based NLM and n-gram LM.

4. Conclusion and Future Work

In this paper we proposed two new models to incorporate morphological information into the SMT pipeline. In the first model we enriched a factored SMT model via a new factor which relies on morphology-aware word embeddings. In our model we focus on Farsi. There are similar models (Zou et al., 2013; Passban et al., 2016a,c) which benefit from word embeddings to improve translation of Farsi and other languages. They train bilingual embeddings but in our model we used monolingual embeddings for the same task. In the second model we tried to manipulate the conventional n-gram LM and recompute the scores of word n-grams with a subword-aware NLM. Both methods are able to effectively improve existing SMT models. For our future work we will develop NMT models which have compatible architectures with MRLs and explicitly benefit from morphological information.

Acknowledgments

We thank the anonymous reviewers, as well as Meghan Dowling and Abigail Walsh for their helpful comments, and the Irish centre for high-end computing (`www.ichec.ie`) for providing computational infrastructures. This research is supported by Sci-

ence Foundation Ireland at ADAPT: Centre for Digital Content Platform Research (Grant 13/RC/2106).

Bibliography

Bird, Steven. NLTK: the natural language toolkit. In *COLING/ACL*, pages 69–72, 2006.

Botha, Jan A and Phil Blunsom. Compositional Morphology for Word Representations and Language Modelling. In *ICML*, pages 1899–1907, Beijing, China, 2014.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*, pages 1724–1734, Doha, Qatar, 2014.

Duh, Kevin, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation. In *ACL (Volume 2: Short Papers)*, Sofia, Bulgaria, 2013.

Hinton, Geoffrey E. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

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

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI-16*, pages 2741–2749, Phoenix, Arizona, USA, 2016.

Koehn, Philipp. Statistical Significance Tests for Machine Translation Evaluation. In Lin, Dekang and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Koehn, Philipp and Hieu Hoang. Factored Translation Models. In *Conference on Empirical Methods in Natural Language Processing Conference on Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, 2007.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NACL*, pages 48–54, Edmonton, Canada, 2003.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *ACL*, pages 177–180, Prague, Czech Republic, 2007.

Li, Peng, Yang Liu, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. A Neural Reordering Model for Phrase-based Translation. In *COLING*, pages 1897–1907, Dublin, Ireland, 2014.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, Pennsylvania, USA, 2002.

Passban, Peyman, Andy Way, and Qun Liu. Benchmarking SMT Performance for Farsi Using the TEP++ Corpus. In *EAMT-15*, pages 82–89, Antalya, Turkey, 2015.

Passban, Peyman, Chris Hokamp, Andy Way, and Qun Liu. Improving Phrase-Based SMT Using Cross-Granularity Embedding Similarity. In *EAMT*, pages 129–140, Riga, Latvia, 2016a.

Passban, Peyman, Qun Liu, and Andy Way. Boosting Neural POS Tagger for Farsi Using Morphological Information. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(1):4:1–4:15, 2016b. ISSN 2375-4699.

Passban, Peyman, Qun Liu, and Andy Way. Enriching Phrase Tables for Statistical Machine Translation Using Mixed Embeddings. In *COLING*, pages 2582–2591, Osaka, Japan, 2016c.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *EMNLP*, Doha, Qatar, 2014.

Salton, Gerard, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Smit, Peter, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *ACL*, pages 21–24, Gothenburg, Sweden, 2014.

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. Highway networks. In *ICML Deep Learning workshop*, Lille, France, 2015.

Stolcke, Andreas. SRILM - an extensible language modeling toolkit. In *INTERSPEECH*, Denver, Colorado, USA, 2002.

Tamura, Akihiro, Taro Watanabe, and Eiichiro Sumita. Recurrent Neural Networks for Word Alignment Model. In *ACL (Volume 1: Long Papers)*, pages 1470–1480, Baltimore, Maryland, 2014.

Toutanova, Kristina, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NACL*, pages 173–180, 2003.

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. *CoRR*, abs/1409.2329, 2014.

Zou, Will Y, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*, pages 1393–1398, 2013.

Address for correspondence:
Peyman Passban
peyman.passban@adaptcentre.ie
ADAPT Centre, School of Computing, Dublin City University, Ireland.