Translating Low-Resource Languages by Vocabulary Adaptation from Close Counterparts

PEYMAN PASSBAN, QUN LIU, and ANDY WAY, ADAPT Centre, School of Computing, Dublin City University, Ireland

Some natural languages belong to the same family or share similar syntactic and/or semantic regularities. This property persuades researchers to share computational models across languages and benefit from high-quality models to boost existing low-performance counterparts. In this article, we follow a similar idea, whereby we develop statistical and neural machine translation (MT) engines that are trained on one language pair but are used to translate another language. First we train a reliable model for a high-resource language, and then we exploit cross-lingual similarities and adapt the model to work for a close language with almost zero resources. We chose Turkish (Tr) and Azeri or Azerbaijani (Az) as the proposed pair in our experiments. Azeri suffers from lack of resources as there is almost no bilingual corpus for this language. Via our techniques, we are able to train an engine for the Az→English (En) direction, which is able to outperform all other existing models.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Neural networks**; *Artificial intelligence;*

Additional Key Words and Phrases: Statistical machine translation, neural machine translation, low-resource languages

ACM Reference Format:
Peyman Passban, Qun Liu, and Andy Way. 2017. Translating low-resource languages by vocabulary adaptation from close counterparts. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 16, 4, Article 29 (September 2017), 14 pages.
DOI: http://dx.doi.org/10.1145/3099556

1. INTRODUCTION

Machine translation (MT) by its nature is a very complicated process, as it has to deal with syntactic, semantic, morphological, and many other types of linguistic complexities at the same time in more than one language. The problem becomes more severe where source and target languages have considerable differences. Pre/post-translation techniques are explored to mitigate these sorts of complexities, for example, if source and target languages have different word-ordering systems pre-reordering is applied to address the problem [Miceli-Barone and Attardi 2013], or if languages are morphologically different, words on the complex (or rich) side are decomposed into simpler subunits to balance the morphological symmetry [Goldwater and McClosky 2005]. Similarly, there are solutions for syntactic [Yamada and Knight 2001] and semantic [Jones et al. 2012] problems.

This research is supported by Science Foundation Ireland at ADAPT: Centre for Digital Content Platform Research (Grant 13/RC/2106).
Authors' addresses: P. Passban, Q. Liu, and A. Way, ADAPT Centre, School of Computing, Dublin City University, Ireland; emails: {firstname.lastname}@adaptcentre.ie.

DOI: http://dx.doi.org/10.1145/3099556

All these solutions are applicable where corpora and tools (annotators, analysers, parsers, etc.) are available. However, for some languages that are referred to as *low-resource languages*, we do not have access to such data and technology. Usually, for these cases processing knowledge is transferred from a high-quality model that relies on a linguistically similar language. This means that a model is trained on one language and then adapted to the language of interest. This approach is called *transfer learning* [Pan and Yang 2010] and has been applied to many research fields, including natural language processing (NLP). Jiang et al. [2015] applied transfer learning to part-of-speech (POS) tagging and dependency parsing. First they train a POS tagger/parser on a high-resource language, then they use the same model to annotate/parse languages that suffer from data scarcity. To this end, they benefit from (i) English as an interlingua to bridge languages, and (ii) word-alignment information to transfer the model to the target language. After transferring they fine-tune the model via a small dataset from the low-resource language. The whole process is called *language and annotation adaptation* [Jiang et al. 2015]. The application of transfer learning is not limited to this example alone, where Wang and Zheng [2015] reviewed different transfer-learning-based models for many NLP tasks. However, models reviewed in their work do not include MT and to the best of our knowledge, this work along with Zoph et al. [2016] are the only models that study transfer learning for the problem of MT.

In this article, we use transfer learning for MT. Domain adaptation [Koehn and Schroeder 2007] is a well-known example of MT transfer-learning, in which models are fine-tuned to particular genres. MT engines cannot perform well when testing conditions deviate from training conditions. The basic idea is to exploit in-domain training data to adapt components of an already trained engine. This is a type of intra-language transfer-learning but we are interested in inter-language models, namely the models that are trained on a pair of languages but applied to other languages. Zoph et al. [2016] proposed a model that exactly applies this idea and our work is based on their model.

The model of Zoph et al. [2016] trains neural engines to translate different languages into English. In their approach, first they select languages such as French or Spanish for which there are plenty of resources available. After training a reliable translation model (*mother*),[1] it is adapted for low-resource languages (*child*). They replace the vocabulary set of the source side in the *mother* model with a vocabulary set of the *child* language. Vocabulary modification changes the *mother* model into another model to translate the *child* language into English. Although translations generated by such a modified model are not necessarily good, the model produced via this procedure is a neural machine translation (NMT) engine for the *child*→En direction, which is a noteworthy achievement. This is the first phase of the procedure. The next phase is to adapt the model for the *child* language and improve its quality. To this end, a small set of bilingual sentences (*child*↔English) are used to optimize the model with respect to properties of the *child* language.

They considerably improved translation quality for their low-resource languages in this way. Language pairs used in their research were randomly selected, which means there may or may not be linguistic and structural similarities between *mother* and *child* languages. However, they demonstrated via experimental investigations that performance improves where *mother* and *child* languages are close to each other. We expand this idea in our research.

In this article, we work on the translation of Azeri, which is a highly agglutinative and morphologically rich language. Moreover, there are almost zero resources for this language with just a handful of Azeri MT systems. These are the main reasons we

[1]X→English.

targeted Azeri. Although Dilmanc[2] [Fatullayev et al. 2008] was proposed as an MT solution for this language, its performance is not satisfactory (see Section 4) and we aim to achieve better results. To translate Azeri, we acquire translation knowledge from an engine trained on Turkish, as these two languages are close to each other (see Section 2) from a linguistic point of view. First, we develop NMT and statistical MT (SMT) engines for Turkish, and then we adapt those models to work for Azeri. Although Turkish is also considered as a low-resource language, there are many more resources available for that language compared to Azeri. Furthermore, Turkish MT has been investigated from different perspectives in recent years [El-Kahlout and Oflazer 2006; Oflazer and El-Kahlout 2007; Avramidis and Koehn 2008; Bisazza and Federico 2009; Yeniterzi and Oflazer 2010; Zoph et al. 2016], so we can rely on an engine trained on Turkish as a *mother* model.

In this article, we report MT results for Azeri and apply deep learning technologies to this language for the first time, which is the main contribution of this work. It is interesting to see the performance of a newly emerging technology for such a language. Deep-learning-based models can outperform conventional MT alternatives for well-studied languages such as English and German [Bentivogli et al. 2016], but there is no result for resource-poor languages such as Azeri. The structure of the article for the next sections is organized as follows. Section 2 briefly describes Turkish and Azeri and discusses why we are interested in applying NMT to these languages. The fundamentals of NMT are also reviewed in the same section. Section 3 describes our proposed models. Section 4 reports experimental results together with our findings, and Section 5 concludes the article with some avenues for future work.

2. BACKGROUND

In this section, we briefly explain Turkish and Azeri, and why we use NMT. We also explain the fundamentals of NMT. Both Turkish and Azeri are Turkic languages that have very similar syntactic and semantic structures. They also share some morphological properties. Each of them can be viewed as a variation of the other one, but this does not mean that these languages are exactly the same. There are fundamental differences between these languages, some of which we briefly mention here. Some of our examples in this section are taken from Öztopçu [1993]:

—**Orthography and Phonology:** Some words have almost the same form in both languages but they vary in some characters, for example the Turkish sound *"k"* is written as *"x"* or *"q"* in Azeri, such as in the Turkish word *"doda***k***"* meaning *"lip,"* which is *"doda***q***"* or *"doda***x***"* in Azeri.

—**Morphology:** Most of the differences between these languages are related to morphological variations, for example, the Turkish suffix *"-miş"* (sounds /mi∫/) is *"-ib"* in Azeri, such as in *"gel***miş***"* and *"gel***ib***,"* meaning *"has come."*

—**Vocabulary:** Although a large set of words is shared between these languages, there are some Azeri words that are either not found in Turkish or they have different meanings, for example, *"sabah"* means *"morning"* in Turkish and *"tomorrow"* in Azeri, or words such as *"subay,"* meaning *"single,"* do not exist in Turkish.

—**Syntax:** Both Turkish and Azeri are SOV (Subject–Object–Verb) languages, but there are some syntactic patterns in Azeri that do not follow this structure. Usually, these cases are less common or do not exist in Turkish, for example, *"I_1 $know_2$ $that_3$ you_4 are_5 $going_6$ to_7 a_8 $conference_9$"* is *"$bilirəm_{1,2}$ ki_3, $konfransa_{7,8,9}$ $gedirsən_{4,5,6}$"* in Azeri and *"$konferansa_{7,8,9}$ $gittiğini_{4,5,6}$ $biliyorum_{1,2,3}$"* in Turkish (showing word alignments with subscripts). There are probably more grammatical exceptions in

[2]http://dilmanc.az/en/translate.

Azeri than Turkish, as Azeri has been affected by the grammar of other languages such as Farsi (Persian) and Russian.

—**Semantic:** There are many phrases and sentences that share the same set of words in both languages but have different meanings in each of them, for example, the translation of *"you look like my sister"* in Azeri sounds like *"you caress my sister"* in Turkish. An interesting point is that when the sentence is translated into these languages, everything is exactly the same for both (words, orders, etc.). Each word even has the same meaning and translation when context is disregarded, but when they appear together they convey different meanings in each of these languages. As another example, if we translate *"the plane will be landing in 10 minutes"* into Azeri, the translation would be completely understandable for native Turkish speakers as it consists of Turkish words but it sounds like *"the plane will crash in 10 minutes"* to them.

As these examples show, these languages are different, but they can still borrow words and linguistic knowledge from each other. Azeri is a language with more than 26 million speakers,[3] but there is a limited amount of research work for this language, which we believe is due to data scarcity. Motivated by such problems, we think that cross-lingual similarities between Turkish and Azeri can enable us to benefit from translation knowledge of Turkish for Azeri, and we decided to use NMT in this regard. In neural approaches, everything is learned through training datasets and there is no need for feature engineering. We only need to manipulate training datasets and the engine itself tries to extract useful information, and it learns the new data distribution. Accordingly, transfer learning is easily applicable in such models [Bengio 2012]. This process is not very straightforward (and sometimes impossible) in conventional statistical models, as in the phrase-based SMT approach [Koehn et al. 2003] everything including translation, language, and reordering models has to be modified, which could even be as expensive as training a brand new model. For these reasons, we prefer to rely on an NMT engine as the main model. However, we propose several SMT alternatives. In the next section, we briefly review the fundamentals of NMT.

2.1. Neural Machine Translation (NMT)

Recently, NMT has emerged as a very powerful alternative for conventional SMT models. Bentivogli et al. [2016] published their experimental results on the comparison of SMT and NMT models and illustrated that for many cases NMT outperforms other models. Similarly, there are other successful neural models such as Chung et al. [2016], which achieved state-of-the-art results.

Neural models are not new in the field as Ñeco and Forcada [1996] proposed such a solution 20 years ago. Other models [Schwenk et al. 2006; Kalchbrenner and Blunsom 2013] have also been proposed thereafter, but for the first time, Cho et al. [2014] and Sutskever et al. [2014] were able to design effective architectures for MT. Such models follow the encoder-decoder architecture. In the encoder-decoder approach two recurrent neural networks (RNNs) are trained jointly to maximize the conditional probability of a target sequence (candidate translation) $y = y_1, \ldots, y_m$ given a source sentence $x = x_1, \ldots, x_n$. Input words are sequentially processed one after another until the end of the input string is reached. An encoder reads words and maps the input sequence into a fixed-length representation. At each time step t, an input word is taken and the hidden state is updated accordingly. This process can be formulated as in Equation (1):

$$h_t = f\big(\mathbb{E}_x[x_t], h_{t-1}\big), \tag{1}$$

[3]According to Wikipedia statistics for 2007 (https://en.wikipedia.org/wiki/Azerbaijani_language).

where $h_t \in \mathbb{R}^d$ is the hidden state (a vector) at the time step t and $f(.)$ is a recurrent function such as long short-term memory (LSTM) [Hochreiter and Schmidhuber 1997] or gated recurrent unit (GRU) [Cho et al. 2014]. $f(.)$ is responsible for updating the hidden state of the layer and other associated units (if there are any, such as memory units, etc.) $\mathbb{E}_x \in \mathbb{R}^{|\mathcal{V}_x| \times d}$ is an embedding matrix for source symbols (d is the embedding size). The embedding matrix is a look-up table whose cells are treated as network parameters and updated during training. The embedding (numerical vector) for the vth word in $\mathcal{V}_x$ (vocabulary) resides in the vth row of the table.

After processing all words in the source sequence, h_n is a summary of the input sequence that is referred to as the context vector (c). Another RNN is initialized by c and tries to generate a target translation. One word is sampled from a target vocabulary $\mathcal{V}_y$ at each time step. The decoder conditions the probability of selecting a target word y_t on the context vector, the last predicted target symbol, and the decoder's state. This can be formulated as in Equations (2):

$$\begin{aligned} y_t &= g(\mathbb{E}_y[y_{t-1}], s_t, c), \\ s_t &= f(\mathbb{E}_y[y_{t-1}], s_{t-1}, c), \end{aligned} \tag{2}$$

where s_t is the decoder's hidden state. Since we compute the probability of selecting y_t as the target word, $g(.)$ should provide a value in the range $[0, 1]$. The most common function for $g(.)$ is *Softmax*. Both encoder and decoder RNNs are trained jointly to maximize the log probability of generating a target translation $\mathbf{y}$ given an input sequence $\mathbf{x}$, so the training criterion can be defined as in Equation (3):

$$\max_{\theta} \frac{1}{K} \sum_{k=1}^{K} \log(y_k | x_k), \tag{3}$$

where θ is a set of network parameters and K indicates the size of the training set.

As previously mentioned, recurrent functions in encoder-decoder models are not normal mathematical functions. Simple RNNs are not powerful enough to capture all information about sequences, so more powerful alternatives such as LSTM RNNs are required. An LSTM unit mitigates the problem of long-distance dependencies by augmenting a simple RNN with a memory vector $m_t \in \mathbb{R}^d$. More formally, an LSTM unit takes x_t, h_{t-1} and m_{t-1} as its input and produces h_t and m_t via the calculations in Equations (4):

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\ g_t &= \tanh(W_g x_t + U_g h_{t-1} + b_g), \\ m_t &= f_t \odot m_{t-1} + i_t \odot g_t, \\ h_t &= o_t \odot \tanh(m_t), \end{aligned} \tag{4}$$

where i_t, f_t, and o_t indicate the input, forget and output gates, respectively. $\sigma(.)$ is an element-wise sigmoid function and W_ρ, U_ρ, and b_ρ, $\rho \in \{i, f, o, g\}$ are all network parameters.

In the basic encoder-decoder architecture all input words contribute equally to the decoding phase (they are all crammed in the hidden layer regardless of their importance and impact on the target word), which is not true in reality. To generate a target word, we know that a set of source words directly affect the decoder, while other words have a weak or zero impact. This important feature does not exist in the basic encoder-decoder architecture. Bahdanau et al. [2014] proposed an attention mechanism to address this

shortcoming. At each time step t, an exclusive context vector is defined with respect to hidden states of both the encoder and decoder. In this model the impact of words is incorporated by assigning weight values. Therefore, Equations (2) can be rewritten as in Equations (5):

$$\begin{aligned} y_t &= g(\mathbb{E}_y[y_{t-1}], s_t, c_t), \\ s_t &= f(\mathbb{E}_y[y_{t-1}], s_{t-1}, c_t), \end{aligned} \tag{5}$$

where c_t is exclusively generated for each time step based on h_t and s_t, and is defined as in Equation (6):

$$c_t = \sum_{j=\hat{t}}^{n} \alpha_{t\hat{t}} h_{\hat{t}}, \tag{6}$$

where $\alpha_{t\hat{t}}$ is a weight value and is defined as in Equations (7):

$$\begin{aligned} \alpha_{t\hat{t}} &= \frac{\exp(e_{t\hat{t}})}{\sum_{j=1}^{n} \exp(e_{tj})}, \\ e_{t\hat{t}} &= s(h_{\hat{t}}, s_{t-1}), \end{aligned} \tag{7}$$

where $e(.)$ is an alignment model that scores the relevance of source words given s_t. $a(.)$ is a combinatorial function modeled via a feed-forward connection. Given the demonstrable benefit of this additional component, the NMT engine used in our experiments is an encoder-decoder model with attention.

3. PROPOSED MODEL

We designed different SMT and NMT models for our experiments. The main focus of the article is to transfer the neural model from the Tr→En direction to the Az→En direction. We also trained a number of SMT models along with our neural engine to provide better comparisons of different settings. Since the SMT architecture is quite clear, we explain only the neural model in this section.

Our NMT model relies on that of Zoph et al. [2016], in which they first train a high-quality translation engine for the French→En setting, then transfer the translation model for a *child*→En direction. To transfer the model, a small set of *child*→En bilingual sentences is used to fine-tune the parameter set of the *mother* model. NMT engines (and deep neural models in general) are data-hungry models, so a massive amount of data is required to set network parameters and reach the final configuration. This is not affordable for languages that suffer from data scarcity. Moreover, training a neural model for sparse datasets is quite challenging. If one side of the translation model belongs to a morphologically rich language or has a very large vocabulary, then it is hard to find a precise configuration between input and output feature spaces (translating the source language into the target language).[4]

To address these problems, Zoph et al. [2016] used an existing *mother* model as a base rather than training a new model based on the *child* language. The intuition behind the model is that the base model provides a prior distribution that preserves useful information from which the *child* NMT engine can start its optimization. In the basic model, the *mother* vocabulary set $\mathbb{E}_x$ is substituted by a vocabulary set of a new *child* model $\mathbb{E}_{new}$ without any assumption and preprocessing (random substitution). This model could work for our case too, but as we focus on similar language pairs, we can benefit from linguistic similarities and reach a better result. In our case, first

[4]Unfortunately, we have to confront all of these problems (less data, sparse data set, and large vocabulary) in Azeri.

we extract the vocabulary sets from training corpora. $\mathcal{V}_{tr}$ and $\mathcal{V}_{az}$ are the vocabulary sets belonging to the Turkish and Azeri languages, respectively. By use of a dictionary (bilingual lexicon), we automatically find word translations between the two vocabulary sets. Any type of dictionary or MT model can be used in this regard,[5] which we use an SMT model (see Section 4 for more details). What we obtain from this procedure is a set of word-level alignments, such as $WT = \{(w_i^{tr} \Rightarrow w_j^{az}); w_i^{tr} \in \mathcal{V}_{tr} \ \& \ w_j^{az} \in \mathcal{V}_{az}\}$, in which w_i^{tr} and w_j^{az} are translationally equivalent. We use these alignments in our transfer process to substitute vocabulary sets in a better way (than a random substitution).

After creating the alignment set WT, we stem all Turkish words. The process is simply denoted with $a \mapsto b$, where b is a stem form of a. Turkish is an agglutinative language, so several affixes could be sequentially added to a stem to generate new variations of the word, but the base meaning stays the same, for example, *"oda"* meaning *"room"* could be extended with additional affixes such as in *"oda+m"* (*my room*) or *"oda+m+da"* (*in my room*), but they all convey the base meaning *"room."* The reason we stem Turkish words is that in the stemmed form the number of unique Turkish tokens is considerably decreased, which means the complexity is mitigated accordingly. In the presence of surface forms, we have to find inflected and precise counterparts of Turkish words from the Azeri vocabulary set (via the alignments), which is impossible for most words. We mentioned that Azeri is a low-resource language, and we are not able to provide such a rich vocabulary for Azeri. Accordingly, by stemming we deal with a limited set of Turkish tokens and it is practicable to find their Azeri peers. Although this simplification is very impactful, this is not the main reason to stem Turkish words.

As previously mentioned, the main motivation behind proposing our model is to feed a *child* model with high-level translation knowledge, which comes from a *mother* model. In such a scenario, gisting could be sufficient, as we do not expect the *mother* model to provide its child with fluent translations. If the goal is to translate Turkish into English, then all inflected forms have their own impact on the final translation and stemming Turkish words degrades performance, while in our approach the *mother* model is (only) supposed to steer us toward a point close to the target translation. Accordingly, an approximate estimation of the word could be enough, for example, in the aforementioned example the *mother* model should be able to generate alternatives such as *rooms*, *my room*, or indeed any other related translations. The main model that is responsible for translating Azeri into English is fed by such information, and based on the inflected inputs (in Azeri) and context, it decides on the surface form of the final translation. This is the main intuition behind stemming. We do not want to confuse/mislead the Azeri model with complex morphological forms of Turkish words. We produce the base translation based on stemmed forms, which conveys high-level meanings and provide approximately correct translations. Then the main Azeri model uses the base translation together with its own knowledge to select the best target word.

After substituting the Turkish vocabulary set with its Azeri version, we use a small set of Az→En sentences to tune the model (*mother*) to transfer its Turkish side to Azeri (*child*). If the Turkish side is based on a large number of morphologically rich words, then it would preserve sparse and quite complicated information. It is hard to transfer such a complex feature space via a small set of Az→En sentences, but when the source side is based on a limited number of tokens with simple stem forms, we can expect the same small set to dominate and transform the existing distribution to its own distribution. The point of having prior Turkish knowledge in the model is (again) because of the

[5]An example of using Google Translate to map Turkish words to Azeri words: https://translate.google.com/#tr/az/erkekler. We perform the same process using our SMT model.

Table I. The First Column Shows the Different Corpora Used in Our Experiments, and the Second Column Shows from Which Collections They Are Selected. T Stands for Tanzil and O Stands for OpenSubtitle2016. The Fourth Column Shows the Size of the Training Set

Corpus	Collection	Language Pair	Size
Corpus_1	T	Az–En	268K
Corpus_2	T	Az–Tr	186K
Corpus_3	T	Tr–En	1.17M
Corpus_4	O	Tr–En	4.50M
Corpus_5	T + O	Tr–En	5.67M

size of the small data set. By such *a priori* distribution, we try to provide an approximately correct initialization point for the Azeri model from which we can start and use the the small data set to reach the final Azeri distribution/configuration. If we were to start from the beginning (an empty encoder with no prior distribution/configuration), then it would be almost impossible to capture such a distribution by use of the small Azeri set (only).

We train the *mother* model for the Tr→En direction with Turkish stems. This means that we do not change the English side but Turkish words are substituted with their stems. Together with the WT set, this is the second change that helps us achieve better results. After training the *mother* model, we replace the source vocabulary set $\mathbb{S}_{tr}$ (set of Turkish stems) with $\mathbb{E}_{az}$. Zoph et al. [2016] substitute the two sets without any assumption, whereas we provide a more informed substitution based on the WT set and stems. Words are represented via vectors in neural models, so we have a vector for each stem in the source side. In the substitution phase, for each Azeri word $w_j^{az} \in \mathcal{V}_{az}$, we select the stem vector of its translation to assign, that is, if $w_i^{tr} \Rightarrow w_j^{az}$, then the vector for w_j^{az} is initialized with σ_i where $w_i^{tr} \mapsto \sigma_i$. In the proposed model, network parameters are trained based on simple stem forms without paying attention to (fine-grained) morphological variations. According to our experimental results this type of substitution helps us converge faster and improves the model's quality.

4. EXPERIMENTS

4.1. SMT Experiments

We perform several experiments to evaluate our models. As previously mentioned, the goal of the article is to design SMT and NMT engines for the Az→En direction. In our experiments, we use different bilingual corpora whose information is provided in Table I.

Corpus_1 is the largest bilingual corpus available for the En–Az pair, which is a part of the Tanzil collection [Tiedemann 2012].[6] Corpus_2 and Corpus_3 are also from the same collection. Tanzil includes the Quran's translations into different languages, so Corpus_1, Corpus_2, and Corpus_3 are from the same text and genre but from different parts. These corpora are clearly related to each other but are not eachother's exact translations. Corpus_4 is a corpus of 4.5M sentences selected from the OpenSubtitle2016 collection[7] Lison and Tiedemann [2016], and Corpus_5 is the combination of Corpus_3 and Corpus_4. The size of the training set for each corpus is reported in Table I and the test and validation sets include 2K and 3K sentences, respectively (for all experiments). The test and validation sets are selected from the same collection to which the training corpus belongs, for example, TestSet_3 and TestSet_4 include 2K sentences,

[6] http://opus.lingfil.uu.se/Tanzil.php.
[7] http://opus.lingfil.uu.se/OpenSubtitles2016.php.

which the first one was selected from the Tanzil collection and the second one from the OpenSubtitle2016 collection. The test and validation sets for Corpus_3 and Corpus_5 are the same.

Using corpora, we train different SMT models. As we previously mentioned, the main focus is on the neural model, and we accompany the main model with other SMT models for comparative purposes. To this end, first we clean and normalize the corpora by our in-house normalizer, as existing models might not support especial encodings of Turkish and Azeri. We replace all digits with a special token *Digit* and convert infrequent words (occurring less than five times) into *UNK*. We also lowercase all words.[8]

We use Moses [Koehn et al. 2007] with its default configuration and 5g language models trained by the SRILM toolkit [Stolcke 2002]. Language models are trained on the target sides of the corpora. We tune translation engines with MERT [Och 2003]. To evaluate our models, we report their performance based on BLEU [Papineni et al. 2002], which is the most frequently used metric in the field. We have a number of different systems[9]: $Az2En_T$ was trained on Corpus_1 for the Az→En direction and provides a BLEU score of **20.16**. Although Corpus_1 is not large enough to train a robust and reliable SMT engine, it yields an acceptable performance. We consider $Az2En_T$ as the main baseline in our experiments and try to find solutions (using Turkish) to outperform its performance.

We have three Tr→En engines, which we wish to fine-tune for Azeri to surpass the existing baseline system. $Tr2En_T$ was trained on Corpus_3 and evaluated on TestSet_3, for which the BLEU score is **23.71**. $Tr2En_O$ is a similar system for the same direction trained on a quite large(r) corpus (Corpus_4) and its performance on its own test set is **27.11**. Both of these systems work for the Tr→En direction, and we have two corpora for this direction. We can combine these two corpora to have a better and more powerful Tr→En model. The result of the combination is $Tr2En_{T+O}$, which performs with a BLEU score of **25.78** (for TestSet_3). This combination was useful and leads to an improvement form **23.71** to **25.78** (+2.07). $Tr2En_{T+O}$ was trained with Corpus_5, fine-tuned with DevSet_3 (validation set of $Tr2En_T$) and evaluated on TestSet_3. The reason we use these test and validation sets for $Tr2En_{T+O}$ is that we tend to benefit from this system (which is the best existing Tr→En model) to improve the translation quality of the Az→En direction. The test set for the Az→En system ($Az2En_T$) belongs to the Tanzil collection, so to tune $Tr2En_{T+O}$, it is better to have test and validation sets from the same data distribution (the same collection).

The main goal behind training different SMT models is to investigate whether we can benefit from a Turkish model for Azeri, so the idea is to use $Tr2En_{T+O}$ instead of (or together with) $Az2En_T$ to provide better results for the Az→En direction. Although we know that Azeri and Turkish are closely related languages, this closeness is not very clear for Moses, because the quality of translation is not as satisfactory as we expected, so the BLEU score obtained by our SMT models is **29.68** for Az→Tr and **32.33** for the opposite direction. This means that Moses treats this pair as two different languages and $Tr2En_{T+O}$ cannot be directly used to translate Azeri. To exploit $Tr2En_{T+O}$, we rely on Turkish as a pivot language. First the Azeri test set (TestSet_1) is translated into Turkish via $Az2Tr_T$ (trained using Corpus_2). The translation generated by this process is referred to as Tr_p. Then, $Tr2En_{T+O}$ is used to translate Tr_p into English.

[8]Basically, our normalizer is almost the same as that of Moses, but because of special characters and encodings, we developed our own variant.

[9]The name of each system includes the acronyms of the source and target languages. We also have a subscript that provides additional information about the collection or the technique used to train the model.

This process can be summarized as follows:

$$\text{Az} \xrightarrow{Az2Tr_T} \text{Tr}_p \xrightarrow{Tr2En_{T+O}} \text{En}.$$

In this setting, Az is TestSet_1, which is translated into Turkish (Tr_p) via $Az2Tr_T$ (shown above the arrow), and $Tr2En_{T+O}$ translates Tr_p into English. The BLEU score obtained by this process is **14.78**.

This pivoting system is not able to surpass the baseline result but it is a valuable achievement as it relies on Turkish to assist the translation process and provides information about properties of Turkish and Azeri. Pivoting is used to translate between two languages (such as L_1 and L_2) by use of another bridge languages (L_3), where there is no bilingual corpus for the L_1–L_2 pair. In the presence of high-quality engines for the $\text{L}_1 \rightarrow \text{L}_3$ and $\text{L}_3 \rightarrow \text{L}_2$ directions, pivoting can improve MT quality [El Kholy et al. 2013], but because our Az→Tr engine ($Az2Tr_T$) is not a very precise model, pivoting could not help too much.

There is another technique that might be more useful than pivoting. Instead of translating the Azeri test set, we can translate the phrase table. $Tr2En_{T+O}$ is a high-quality engine for the Tr→En direction. If we can manipulate its phrase table and tune it for Azeri, then it could perform well for the Az→En direction too. The phrase table of $Tr2En_{T+O}$ includes bilingual Tr–En phrases. Using $Tr2Az_T$ (trained on Corpus_2), we translate Turkish phrases into Azeri. This part could be interpreted as a sample of transfer learning for a Tr→En model. To transfer an SMT model, we need to manipulate (*i*) the translation function, (*ii*) the language model, and (*iii*) the reordering function. The word-ordering system is the same for Azeri and Turkish, so we do not need to change anything in this regard. Similarly, since the target language is always English in our experiments, we do not need to change language models. We only train a new language model with the English sides of Corpus_1 and Corpus_5 to obtain better results. So far, we have all the required components (word reordering and language modeling) but the translation function. To transfer the translation function, we translate the phrase table. We use $Tr2Az_T$ and translate each Turkish phrase in the phrase table (of $Tr2En_{T+O}$). The system obtained from this process outperforms the results produced by pivoting with a BLEU score of **18.12**. This system is referred to as $Az2En_{ph.t}$. With different combinations and trying several configurations, we could gradually improve the performance for Az→En, but these combinatorial (hybrid) models are still not able to outperform the baseline model.

We have $Az2En_T$ for Az→En, which is our main model and can be viewed as an in-domain model. We also have an additional model ($Az2En_{ph.t}$) for the same direction, which relies on Turkish and was trained on a different corpus. This second engine can be viewed as an out-of-domain model. There are SMT techniques to combine multiple translation systems to benefit from the strengths of the all models. Through such a combination, we can generate a better translation as the out-domain system can provide something useful that is missed in the in-domain model. We combine $Az2En_T$ and $Az2En_{ph.t}$ using the fill-up[10] technique [Bisazza et al. 2011]. The result of this combination is $Az2En_{fl.u}$, which improves $Az2En_T$ by **+0.53** BLEU points. Table II summarizes all the aforementioned configurations.

Based on the BLEU scores reported in Table II, $Az2En_{fl.u}$ has the best performance, which benefits from both Turkish and Azeri. We also report the performance of Google translate (GT) and Dilmanc (rule-based) for the same test set, which perform significantly worse than ours.

[10]http://www.statmt.org/moses/?n=Advanced.Models#ntoc7.

Table II. Different Non-neural MT Models for the Az→En Direction. Improvements Reported for the Systems Are Statistically Significant According to Paired Bootstrap Re-sampling [Koehn 2004] with $p = 0.05$

	System	Description	BLEU
1	*Az2En*$_T$ (baseline)	Az→En (Corpus$_1$)	20.16
2	*Pivoting*	Az $\xrightarrow{Az2Tr_T}$ Tr$_p$ $\xrightarrow{Tr2En_{T+O}}$ En	14.78
3	*Az2En*$_{ph.t}$	Phrase-table translation	18.12
4	*Az2En*$_{fl.u}$	Fill-up (*Az2En*$_T$ and *Az2En*$_{ph.t}$)	**20.69**
5	*Google Translate*	Az→En	14.20
6	*Dilmanc*	Az→En	10.00

4.2. NMT Experiments

We tried different SMT configurations and were able to improve the Az→En baseline model. We are interested in studying the same problem in the field of NMT and investigate whether we can propose an Azeri model based on Turkish. Because of the flexible architecture of NMT models, we believe that it is possible and even more feasible to propose such a solution.

First, we train a model using Turkish stems and English words for the Tr→En direction ($mother_{tr}^{stm}$). Then, we use a set of Az–En sentences to transfer $mother_{tr}^{stm}$ to work for the Az→En direction ($child_{az}^{stm}$). For the first part, we use Corpus$_4$ and the second part Corpus$_1$. In all of our experiments, we use a two-layer encoder-decoder model with attention. The size of the LSTM units is 1000 and the mini-batch size is 128. We have a dropout layer with $p = 0.2$ on the decoder side. The initial learning rate is set to 0.5 with a decay rate of 0.5. We change the learning rate if the development perplexity does not improve. The network was trained using Adam [Kingma and Ba 2015]. To fine-tune $mother_{tr}^{stm}$, we re-train it with Az→En sentences for 100 epochs. We clip the gradient when the gradient norm is greater than 1.0. All parameters are initialized in the range $[-0.08, +0.08]$. To stem Turkish words, we use a stemmer[11] designed on finite state machines [Eryigit and Adali 2004].

For $\mathbb{E}_{en}$, we select all unique English words (types) from the En–Az dataset along with the 20K most frequent types in the En–Tr set (40K in total). For $\mathbb{E}_{az}$, we select all Azeri types (25K), and for $\mathbb{E}_{tr}$, we keep the 50K most frequent types. The $mother_{tr}^{stm}$ model finds a mapping from $\mathbb{S}_{tr}$ (Turkish stem) to $\mathbb{E}_{en}$ (English word). In the transfer phase, we initialize $\mathbb{E}_{az}$ with $\mathbb{S}_{tr}$ by using WT (see Section 3). If we cannot find a counterpart for any member within $\mathbb{E}_{az}$, then we initialize it with a *null* embedding (specific embedding trained during training). The BLEU score for $mother_{tr}^{stm}$ is **29.17** and its child model ($child_{az}^{stm}$) performs with **21.93**, which is a better result than those of *Az2En*$_T$ (baseline) and *Az2En*$_{fl.u}$ (best system).

In $mother_{tr}^{stm}$, we stemmed Turkish words, because it helps us achieve a better result. Before stemming, we had 510K types in the Turkish corpus, which is almost 2.5 times bigger than that of the stemmed corpus (197K). As we have described, stemming considerably mitigates the complexity. Without any pre-processing (no stemming), we selected the top-50K words from the Turkish side and trained a new neural model ($mother_{tr}^{wrd}$). The BLEU score for $mother_{tr}^{wrd}$ is **28.21** and the child model ($child_{az}^{wrd}$) trained based on $mother_{tr}^{wrd}$ obtains a BLEU score of **21.11**, which shows our stemming technique is useful.

So far, $child_{az}^{stm}$ performs better than our other models, but there is a way to improve it further. We apply one additional fine-tuning step to provide a better prior distribution in $mother_{tr}^{stm}$. After training $mother_{tr}^{stm}$, first we tune[12] the model (first tuning) with the

[11] https://github.com/otuncelli/turkish-stemmer-python.

[12] In our setting, tuning means retraining the existing model with a new dataset for 100 epochs.

Table III. Results from NMT Engines for the Az→En Direction

	System	Description	BLEU
1	$child_{az}^{stm}$	based on $mother_{tr}^{stm}$	21.93
2	$child_{az}^{wrd}$	based on $mother_{tr}^{wrd}$	21.11
3	$child_{az}^{dbl}$	using $mother_{tr}^{stm}$ with double tuning	**22.30**

Table IV. The Best SMT and NMT Results for the Az→En Direction

Approach	System	BLEU
Baseline	$Az2En_T$	20.16
SMT	$Az2En_{fl.u}$	20.69
NMT	$child_{az}^{dbl}$	**22.30**

Turkish version of the `Tanzil` data set ($Corpus_3$). $mother_{tr}^{stm}$ is a model based on the `OpenSubtitle` collection. By tuning the same model with the `Tanzil` collection, we try to change its distribution and push it closer to that of `Tanzil`. The model would then be a Tr→En model, which is "aware of" concepts (words, syntax, semantic, etc.) existing in `Tanzil`. Then, we use the Azeri set to perform the second round of tuning. The Azeri set also comes from the `Tanzil` collection. By our double-refinement process, first we try to transfer the `OpenSubtitle`-based Turkish model (Turkish model trained on `OpenSubtitle`) to its `Tanzil`-based version (Turkish model trained on `Tanzil`). We then substituted the Turkish model with an Azeri model from the same data distribution (Azeri model trained on `Tanzil`). In the previous configuration, we changed the language and domain at the same time (one pass of tuning), but in this configuration, first we change the domain and then the language of the model. This double-stage mechanism is able to provide a better model ($child_{az}^{dbl}$) with a better BLEU score of **22.30**, **0.37** points higher than **21.93**. Results for our neural models are reported in Table III and the best results from the all configurations are summarized in Table IV.

5. CONCLUSION AND FUTURE WORK

In this article, we reported several SMT and NMT configurations for translating from Azeri into English. Since Azeri is a low-resource language, we trained models on Turkish and fine-tuned them for Azeri. Priori to our investigation, we were expecting that as Turkish and Azeri are so close to one other, we ought to be able to easily benefit from Turkish models for Azeri; in the worst case, we thought we might only need slight adaptations to use Turkish for Azeri. However, this did not happen in practice and Moses could not recognize the relation/closeness between these languages. We tried to translate these languages into each other, but despite their having lots of linguistic similarities (word order, words, grammatical structures, etc.), the translation quality for this pair was not very good. This fact shows that the similarity is obvious for us (human user) but it should be encoded for the machine. Accordingly, we could not directly use either Turkish or Azeri to boost the translation model of the other one and had to propose SMT and NMT techniques to make a bridge between Turkish and Azeri models. We applied pivoting and translated Turkish phrase tables to be used for Azeri. We combined training corpora and even translation models to achieve better results. We used the fill-up technique and improved the baseline model for the Az→En direction.

Apart from our SMT models, we also trained some neural models and applied transfer learning and vocabulary adaptation. The base NMT model was trained on a Tr–EN corpus and converted to an Az→En model through a tuning phase. The NMT model trained by our technique is able to provide the best result of all systems evaluated. This is the first time that an NMT model has been proposed and evaluated for Azeri. However, our models are not limited to these languages and can be used for any other

closely related pairs. For future work, we plan to make better connections between Turkish and Azeri. In our models, we focused more on MT architectures, whereas it is possible to propose solutions to better connect close languages and transfer close structures. We also plan to expand our work for other NLP tasks apart from MT; for example, it is possible to rely on close languages and propose taggers or parsers for Azeri.

ACKNOWLEDGMENTS

We thank our reviewers and Ireland's high-performance computing centre (https://www.ichec.ie/) for providing computational resources for this work.

REFERENCES

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proceeding of the the Annual Meeting of the Association for Computational Linguistics (ACL'08)*. 763–770.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Unsupervised and Transfer Learning Workshop*. 17–36.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: A case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 257–267.

Arianna Bisazza and Marcello Federico. 2009. Morphological pre-processing for Turkish to English statistical machine translation. In *Proceedings of the 6th International Workshop on Spoken Language Translation (IWSLT'09)*. 129–135.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of the 8th International Workshop on Spoken Language Translation (IWSLT'11)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1724–1734.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1693–1703.

Ilknur Durgar El-Kahlout and Kemal Oflazer. 2006. Initial explorations in English to Turkish statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*. 7–14.

Ahmed El Kholy, Nizar Habash, Gregor Leusch, Evgeny Matusov, and Hassan Sawaf. 2013. Selective combination of pivot and direct statistical machine translation models. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*. 1174–1180.

Gülsen Eryigit and Eref Adali. 2004. An affix stripping morphological analyzer for turkish. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*. 299–304.

Rauf Fatullayev, Ali Abbasov, and Abulfat Fatullayev. 2008. Dilmanc is the 1st MT system for azerbaijani. In *Proceedings of the 2nd Swedish Language Technology Conference (SLTC'08)*. 63–64.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 676–683.

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

Wenbin Jiang, Yajuan Lü, Liang Huang, and Qun Liu. 2015. Automatic adaptation of annotations. *Comput. Linguist.* 41, 1 (2015), 119–147.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of the 24th International Conference on Computational Linguistics*. 1359–1376.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1700–1709.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR'15)*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 388–395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. 48–54.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*. 224–227.

Pierre Lison and Jrg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*. 923–929.

Antonio Valerio Miceli-Barone and Giuseppe Attardi. 2013. Pre-reordering for machine translation using transition-based walks on dependency parse trees. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. 162–167.

RP Ñeco and Mikel L Forcada. 1996. Beyond mealy machines: Learning translators with recurrent neural networks. In *Proceedings of the World Conference on Neural Networks*. 408–411.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics—Volume 1*. 160–167.

Kemal Oflazer and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*. Prague, Czech Republic, 25–32.

Kurtulus Öztopçu. 1993. A comparison of modern azeri with modern turkish. *Azerbaijan Int.* 1, 3 (1993).

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 311–318.

Holger Schwenk, Daniel Dchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*. 723–730.

Andreas Stolcke. 2002. SRILM - An extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP'02—INTERSPEECH)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Confrence on Advances in Neural Information Processing Systems (NIPS'14)*. 3104–3112.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'12)*. 2214–2218.

Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA'15)*. 1225–1237.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*.

Reyyan Yeniterzi and Kemal Oflazer. 2010. Syntax-to-morphology mapping in factored phrase-based statistical machine translation from English to Turkish. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. 454–464.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Received November 2016; revised March 2017; accepted May 2017