Translation of Unknown Words in Low Resource Languages

Biman Gujral bgujral1@jhu.edu
Huda Khayrallah huda@jhu.edu
Philipp Koehn phi@jhu.edu
Department of Computer Science, Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218, USA

Abstract

We address the problem of unknown words, also known as out of vocabulary (OOV) words, in machine translation of low resource languages. Our technique comprises a combination of methods, inspired by the common OOV types observed. We also design evaluation techniques for measuring coverage of OOVs achieved and integrate the new translation candidates in a Statistical Machine Translation (SMT) system. Experimental results on Hindi and Uzbek show that our system achieves a good coverage of OOV words. We show that our methods produced correct candidates for 50% of Hindi OOVs and 30% of Uzbek OOVs, in scenarios that have 1 and 3 OOVs per sentence. This offers a potential for improvement of translation quality for languages that have limited parallel data available for training.

1 Introduction

Key factors in the performance of Statistical Machine Translation (SMT) systems are the volume and domain of available parallel data. Low resource languages lack sufficient amounts of parallel data as well as advanced linguistic tools for analysis. This results in a high percentage of out of vocabulary (OOV) words (unknown words, not seen in the parallel data). The default solution is to copy the unknown word in the translated output, which may work for named entities but only if the two languages share a script.

We propose a system for generating translation candidates for unknown words. The system uses a combination of methods when translating from a low resource language into English. Each method targets different types of unknown words. These can be named entities, borrowed words, compound words, spelling or morphological variants of seen words or content words unrelated to any seen word. We do not employ any language-specific tools to ensure that our system is applicable across all languages, even resource poor ones that may not have specialized tools. New methods can be added or existing ones pruned, based on the properties of the source language. In our current system, we use (i) transliteration, (ii) Levenshtein distance-based search, and (iii) Canonical Correlation Analysis on word embeddings to generate translation candidates.

We also design an evaluation technique to measure the number of unknown words whose correct translation is found within the set of generated candidates. In addition, we propose strategies to integrate these candidates into a SMT system: i) we create a secondary phrase table comprising unknown words and their candidates, and ii) we pre-specify these translation options as markup in the input, for the language model to choose from.

This paper is organized into five sections. Section 2 discusses prior work that addresses

the problem of unknown words, as well as translation of low resource languages and specific domains. We discuss our techniques to generate translations for unknown words as well as their evaluation and integration in Section 3. In Section 4, we present the specific empirical settings used as well as the results and analysis of our experiments. We conclude with a discussion of future work to further address this problem in Section 5.

2 Prior Work

Several methods have been proposed to address the unknown words problem. Many of them are based on generation of new translation pairs from monolingual data in the two languages. Irvine and Callison-Burch (2013) induce new translation pairs from monolingual corpora by modeling it as a supervised classification problem which predicts if a given pair of words are translations of each other based on context, timestamps, frequency, topic and orthography features. They show results in both high and low resource settings. In addition to new lexical translations, there has been work in adding new phrases, induced using lexical reordering, idiom extraction and unknown words as part of seen contexts (Zhang and Zong, 2013).

Habash (2008) handles OOVs in Arabic-English translation by augmenting the phrase table with new entries based on morphological analysis, transliteration, spelling correction and dictionary lookup. Habash and Metsky (2008) present another technique for Urdu-English, wherein they match OOVs to their seen morphological variants to find possible translations. But, these methods are heavily dependent on language-specific resources and linguistic properties, and cannot be directly extended to other languages. In contrast, our system is independent of the language pair. In addition to language-specific tasks, Banerjee et al. (2012) classify OOVs for domain-specific technical support forum parallel data into terminology, spelling errors, content words, URLs, email addresses, and fused words. They use a separate technique to handle each type of OOV including regular expression followed by post-editing, using supplementary parallel data and spell checker. This limits its application to technical domain only and, as with language-targeted techniques, is not broadly applicable.

Vector space models are created and applied in various ways to find semantic similarities. Daumé III and Jagarlamudi (2011) use Canonical Correlation Analysis (CCA) on German-English data to mine translations of OOVs in the new domain. They use contextual and orthographic feature vectors. Faruqui and Dyer (2014) show that bilingually correlated word vectors obtained using CCA perform better than monolingual vectors on word similarity tasks. They use Latent Semantic Analysis on word co-occurrence matrices as well as Skip-gram and RNN based methods from Mikolov et al. (2013a,c) to create these vectors. However, they do not apply this to machine translation.

In this work, we design techniques that can generate translation options for OOV words, irrespective of the source language. We do not use language-specific tools and leverage monolingual data in lieu of additional parallel data, making it suitable for low resource languages. Also, we propose ways to integrate this with an SMT system as well as an evaluation technique to measure the quality of candidate translations generated.

3 Methodologies

This section surveys common types of OOVs and suggests strategies that target different types.

3.1 Types of Unknown Words

We categorize OOV words based on their properties into the following types:

- **Named Entities:** These refer to names of people, organizations, places, etc., that often remain the same across languages. If the writing system for the two languages differ, they

require transliteration.

- **Acronyms:** These are often transliterations and can be considered a subset of named entities. We create a separate category for these due to their distinctive capitalization and punctuation.

- **Borrowed Words:** These are words of the target language that are borrowed in to the source language. They are distinct from names because they are common words that appear in colloquial use of the language, despite having a designated word in the source language. For instance, the word *shirt*, which appears as a transliteration: शर्ट in Hindi. These may also be borrowed words due to the lack of an equivalent word in the source language, for instance, *technology* or *mobile*. We assign them a separate category because they cannot be identified through standard named entity recognition techniques.

- **Borrowed Words with Source-side Inflection:** It is occasionally the case that transliterated English words are combined with the source-side inflection, for instance, to make them plural. These words require a combination of transliteration and morphological analysis to be translated correctly.

- **Source Content Words:** These are words of the source language that are neither borrowed nor named entities. They are not necessarily rare words and appear as OOVs because they did not happen to be seen in the training data. Therefore, this category constitutes a larger percentage of OOVs in low resource settings, which lack training data. This category also includes OOV words whose morphological variants were seen in training data.

- **Misspellings and Typos:** This category comprises words that have multiple spellings, or were typed incorrectly. These words often differ by a character or two from their correct spelling or variant with a known translation. We can translate them by using techniques such as edit distance between these OOVs and the words in training data.

- **Numbers:** These are copied across languages with a few exceptions where the text may have numbers in source language's script.

Borrowed words, content words and named entities are common across many languages. New categories specific to the source language in use can be added.

3.2 Translation Methods

In this section, we describe the methods used to generate translation options for unknown words. These methods are inspried by the different types of unknown words described in Section 3.1.

Levenshtein Distance: This method targets translation of unknown words that have seen morphological or spelling variants in the training data. Levenshtein distance (Levenshtein, 1966) measures the similarity between two strings based on the number of deletions, additions and substitutions required to transform the first string, w_1 into the other, w_2.

We obtain an aligned bilingual lexicon created on the parallel training data using the Moses SMT system (Koehn et al., 2007). For each Hindi word, we choose the English word to which it has been aligned with the maximum probability in the lexicon. Next, we compute Levenshtein distance between the OOV word and every source side word in the parallel corpus. If a close match is found between an OOV and a source side word, its aligned English word is listed as that OOV's candidate translation. We design a few variants of this method:

- **Full words:** This is the straightforward application as explained above. Here, the distance is measured between the full OOV and source side words. The candidates produced by this method are a superset of the other methods.

- **Suffix:** This is implemented to target morphological variants in particular and is suited for languages with suffix-based morphology. It only measures the distance between suffixes of the two words. The length of suffix that is compared can be varied based on properties of the language.
- **Prefix:** This is analogous to the suffix-based method and is added for languages that have a prefix-based morphology system.
- **Vowels only:** This method requires that the consonants in the two words be the same and only the vowels may differ. It is specifically added to target our use case for Hindi, which tends to have spelling variants based on differences in vowels, especially when writing borrowed English words. This method is dependent on obtaining the language-specific vowel set from the user (for English, this would be {*a, e, i, o, u, y*}). In addition to vowel-based differences, this method can also be used to capture language-specific common spelling errors by adding those letters instead of the vowel set.

Word Embedding: We create word embeddings from a combination of parallel and monolingual data using the Continuous Bag of Words model in *word2vec* (Mikolov et al., 2013a,b,c). This finds representations for words in a continuous space such that words similar in meaning are closer in this space than others. This helps capture semantic similarities. After obtaining word embeddings for both languages, we use Canonical Correlation Analysis (CCA).

CCA is a technique used to learn common features between two sets of data or multiple views of a dataset (Hotelling, 1936). That is, it aims to find the common subspace that maximizes the correlation between them. This can be mathematically written as: given two data matrices, $x \in \mathbb{R}^{d_x \times N}$ and $y \in \mathbb{R}^{d_y \times N}$, it finds vectors $u \in \mathbb{R}^{d_x \times 1}$ and $v \in \mathbb{R}^{d_y \times 1}$ such that:

$$\max \frac{\text{covar}(u^\intercal x, v^\intercal y)}{\sqrt{\text{var}(u^\intercal x)}\sqrt{\text{var}(v^\intercal y)}}$$

$$\text{which can be written as: } \max \frac{\mathbb{E}_{x,y}[u^\intercal x, v^\intercal y]}{\sqrt{\mathbb{E}_x[u^\intercal x]}\sqrt{\mathbb{E}_y[v^\intercal y]}}$$

Since the above expression is affine invariant, it can be written as the following constrained optimization problem of finding a k-dimensional subspace such that $U \in \mathbb{R}^{d_x \times k}$ and $V \in \mathbb{R}^{d_y \times k}$:

$$\max \mathbb{E}_{x,y}[\text{trace}(U^\intercal x y^\intercal V)]$$

$$\text{subject to: } \mathbb{E}_x[\text{trace}(U^\intercal x x^\intercal U)] = I_k;\ \mathbb{E}_y[\text{trace}(V^\intercal y y^\intercal V)] = I_k$$

Solving the above gives final projection vectors as $U = \text{top } k \text{ eigenvectors of } C_{xx}^{-1} C_{xy} C_{yy}^{-1} C_{yx}$ and $V = C_{yy}^{-1} C_{yx} U$. Thus, we obtain a matrix that can be used to project data from the two views into a common maximally correlated space.

To apply CCA, we first find words aligned to each other from the training data using GIZA++ (Och and Ney, 2003) and use it to obtain projection vectors for the source and target languages (Faruqui and Dyer, 2014). We use these vectors to project OOV words and a large English corpora into the shared space and compute cosine similarity between an OOV and each English word to find the closest translation candidates. In addition, we also find the cosine similarity between the OOV and the source language words from the parallel corpus, using the aligned English word as the candidate. We pick the top 10 words with the highest cosine similarity using both these techniques, generating 20 candidate translations in total.

This method is beneficial, especially for low resource settings, since it leverages large monolingual corpora, which are more readily available, particularly in the target language. Also, it

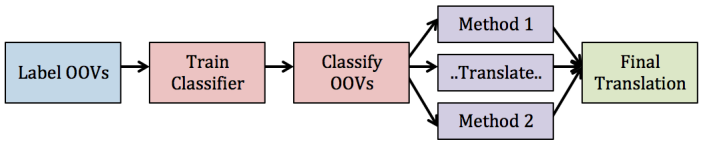

Figure 1: Select method using classifier

has a broad span and helps capture new words without any limitation on the category of OOV (morphological variant, borrowed word etc).

Transliteration: This method is used to translate named entities and words that are borrowed from English. We use the unsupervised transliteration module (Durrani et al., 2014b) in the Moses toolkit. First, a non-transliteration model is learned for source-target aligned words that are not transliterations of each other. Second, a transliteration model is learned for word pairs that are transliterations and can be used for learning alignments at the character level. These character alignments are learned using expectation maximization algorithm and is completely unsupervised. Using these learned alignments, unknown source words are transliterated. To improve the unsupervised transliterations learned by the model, we train a larger character-based language model on the target side. This helps the module to produce more accurate spellings.

3.3 Oracle Performance

Unknown word resolution is divided in two tasks: (i) the generation of translation candidate and (ii) the selection of the correct one. To assess the performance of the first step, we apply each translation method to generate translation candidates for each OOV. We then search for these candidate translations anywhere in the corresponding English reference sentence. We do not use word-level alignments to check for a particular word in the reference sentence due to noisy alignments. This may lead to some false positives, but a brief manual analysis shows that majority of matches are, indeed, correct translations. To avoid further over-counting, we remove all stop words among the candidate translations. This method helps evaluate the quality of candidates produced by each method by providing an upper bound on the number of OOV words that have a correct translation option and can be correctly translated by the SMT system, when these methods are integrated.

In addition to the candidates generated by each method, we add their synonyms obtained from WordNet (Miller, 1995) using NLTK (Bird, 2006). We include synonyms because the candidates obtained through word embeddings tend to produce semantically similar words which may or may not exactly match the ones in the reference sentence. Similarly, Levenshtein distance produces translations from data seen in training, but it is possible that the correct translation may be its synonym. It also helps cover translations that only differ in grammatical number, truecasing, and other minor variations.

3.4 Integration Methods

We consider three ways to integrate our system with an SMT pipeline.

Classifier: This method aims to classify OOV words into the categories described in Section 3.1. Using this classification, one can use the translation method appropriate for the category of that OOV word. For instance, if the word is a misspelling, one can use Levenshtein distance to translate it. We hand-tagged OOV words into these categories and used SVM as our supervised classifier. Features included POS of the OOV word and that of its context (window=3). In addition, we added binary features based on if the lemma of OOV is seen in the source side lemmas (suggesting seen morphological variants), if lemma is same as the OOV word (suggesting a named entity) and if the OOV is just comprised of numbers. Length of the word was also used. The entire proposed pipeline for this method is shown in Figure 1.

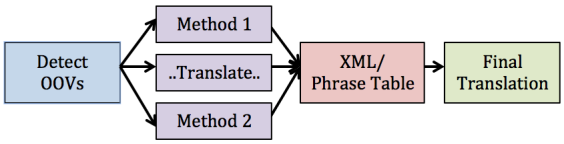

Figure 2: Use all translation methods

XML Markup: Here, we use each translation method proposed in Section 3.2 to generate translation options for each OOV. We then add the generated candidates as XML-markup around the OOV in the source test file. Moses has the ability to use such externally-provided translations while decoding (Koehn and Haddow, 2009). Using this method, we use all translation methods proposed in Section 3.2 to generate translation options for each OOV but we do not add any scores to the candidates. The target-side language model chooses the candidate for each OOV.

Secondary Phrase Table: To avoid basing the decision of picking the best candidate entirely on the language model, we implement an additional technique. As with XML Markup, we first apply all the translation methods to generate candidates for every OOV. We then create a secondary phrase table comprising OOVs exclusively. For every entry in the phrase table, we use three binary features to indicate the method used to generate the translation candidate. We use one feature to indicate the cosine score for candidates generated using word embeddings. For candidates generated using Levenshtein distance, we use that distance as a feature. In addition to these, we also experimented with using inverse frequency of the candidate word in a large monolingual English corpus as a feature. This is added to balance the preference of language model to always choose the most frequent word as the correct translation. Another variant of secondary phrase table that we implemented discards multi-word candidates. This method is included to prune the candidate list because single word OOVs are more likely to have single word translations.

4 Experiments

In this section, we discuss the data and experimental settings we use for implementing and evaluating our translation methods and the obtained results.

4.1 Data

We use Hindi as the language for our experiments because of its relatively low resource nature, rich morphology and knowledge of the language. We use Hindi-English news data from the Workshop on Statistical Machine Translation 2014 (Bojar et al., 2014).[1] Details of the source side of this parallel data are listed in Table 1. There are about 2500 sentences in the test set and about as many OOV tokens. Thus, there is a considerable percentage of unknown words.

We also run experiments on Uzbek-English. This data is made available by the Linguistic Data Consortium (LDC2015E89). The training, tuning and test data are sampled from a combination of Uzbek data obtained from news, Wikipedia, social media and discussion forums and translated into English. Additionally, there is Uzbek-English news text that was published in both languages. Details of dataset size are given in Table 1. This setting is lower resource than Hindi. The effects of reduced parallel data can be seen in the analysis. There are about 1000 sentences in the test set and the average number of OOVs per sentence is 4, compared to only 1 for Hindi.

4.2 Translation Methodologies

We describe the implementation details of each translation method in this section.

[1] http://statmt.org/wmt14/translation-task.html

Data	Hindi			Uzbek		
	Types	**Tokens**	**Sentences**	**Types**	**Tokens**	**Sentences**
Train	117k	3.5m	274k	49.3k	161.4k	55k
Tune	2.5k	9.2k	520	7.2k	15.2k	1k
Test	8.7k	49k	2.5k	7k	15k	1k
OOVs(Test Set)	1.9k	2.9k	-	3.7k	4.8k	-

Table 1: Details of source-side in the parallel data

Method	OOV Types w/ atleast 1 Candidate Generated		OOV Types w/ Correct Candidates Detected	
Vowel-based	1025	51.9%	96	4.9%
Suffix-based	1020	51.6%	211	10.7%
Word-based with distance<=1	1210	61.3%	289	14.6%
Word-based with distance<=2	1651	83.6%	475	24.1%

Table 2: Comparative performance of variants of Levenshtein Distance (Hindi)

Levenshtein Distance: We consider distances of less than or equal to 2 for matching words since Hindi does not have compounding property or unusually long words. For words smaller than length 4, we consider only a distance of 1 to avoid excessive incorrect matches. The same settings are used for Uzbek.

The variants of Levenshtein distance based on suffixes and vowels are beneficial when the goal is to focus on morphological variants or spelling variants respectively. We see good performance by using Levenshtein distance on full words Hindi, this is likely because Hindi is both morphologically rich and prone to spelling variants due to the high percentage of borrowed English words. This is further confirmed by Table 2. We find that using a distance limit of 2 increases the search space but also leads to a considerable improvement in performance, making it a worthy trade-off.

Word Embedding: For this technique, we collect Hindi monolingual data from Wikipedia dump (Al-Rfou et al., 2013) and Commoncrawl (Buck et al., 2014),[2] with a total of about 29 million tokens. For Uzbek, the monolingual data is sampled from the same data from which the parallel training data was sampled. For English, we used the Wikipedia data with about 127 million tokens.[3] To the source side monolingual dataset, we add the source side of train, tune and test sets from the parallel data, since embeddings must be generated for OOVs. For English, we only add the target side of the parallel data to the monolingual corpus and not the tune and test reference sentences.

We use *gensim*, Python library's *word2vec* module (Řehůřek and Sojka, 2010) to create word embeddings for each monolingual corpus. We use the Continuous Bag of Words model and vectors of length 100. For both Hindi and Uzbek, we use a low *min_count* setting of 2, such that it only filters words with frequency less than 2. We set this count low in order to obtain embeddings for as many OOVs as possible. Due to the large size of the English corpus, we set this count to 10 to obtain good embeddings and maintain accuracy of the candidate translations. Table 3 shows the number of OOVs for which an embedding is obtained for different *min_count* values in Hindi. The third column shows the oracle performance i.e. the maximum number of OOVs for which the correct candidate was generated using word embeddings.[4] As expected, increasing the minimum frequency filters more OOVs, causing the number of OOVs with embeddings to decrease. But, the number of OOVs with correct candidates do not decrease sharply. This shows that the OOVs seen sufficiently high number of times in monolingual data tend to be more

[2] `http://statmt.org/ngrams/`

[3] `https://code.google.com/archive/p/word2vec/`;`http://mattmahoney.net/dc/textdata.html`

[4] Calculated using the method in Section 3.3

Min Count	OOV Types w/ atleast 1 Candidate Generated	OOV Types w/ Correct Candidates Detected
2	1104	169
4	945	144
6	828	151
8	743	125
10	681	125

Table 3: Effect of varying minimum frequency attribute in *word2vec* training for Hindi

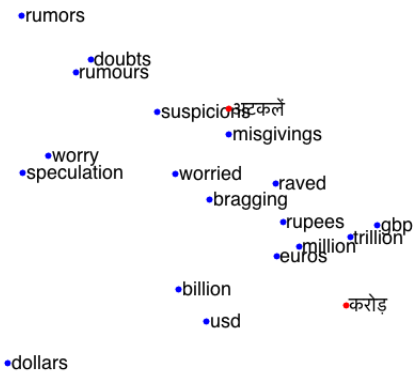

Figure 3: 'अटकलें' (*to speculate*) and 'करोड़' (*crore/ten million*); both have neighboring English words with similar meanings closer to them.

accurately translated due to better vector representations.

Figure 3 shows the top ten words obtained for two Hindi OOVs through cosine similarity between projected OOVs and projected English monolingual corpora.[5] Note that करोड़ which means *crore* (ten million in Indian numbering system) is close to related words like *million, billion, rupees* and so on and farther from *speculation, worried* etc.

Transliteration: As described in Section 3.2, we use Moses to transliterate all the OOV words (Durrani et al., 2014b). We use the monolingual English news corpora from past WMT shared tasks (2007-2012) [6] to create the larger character-based language model for more accurate spellings. This data has 1.5 billion words as opposed to 2.9 million words in the target side of the parallel corpus. Since Uzbek uses a Latin script, we copy the OOV for transliteration.

4.3 Evaluation

We present results of the oracle evaluation as discussed in Section 3.3.

Oracle Performance: The methods differ in how many candidates they produce for each word. We limit transliteration to 1 candidate and the word embeddings method to produce 20 candidates. Levenshtein distance method produces 18 candidates on average. Note that if an OOV word does not have related words (morphological and spelling variants) seen in the source side of training data, the Levenshtein distance method may fail to find the correct candidates or any candidates at all. Similarly, *word2vec* is set up such that every word must appear at least twice in the monolingual data for it to have a vector representation. This may lead to some OOVs being filtered. Transliteration produces a translation candidate for each OOV.[7]

We report the number of OOV types that obtain at least one candidate translation from our

[5] We plot the first vs fifth dimensions here for visual clarity. The candidates shown are the actual top 10 candidates as obtained through a cosine distance across all dimensions

[6] `http://www.statmt.org/wmt12/`

[7] The number of OOVs transliterated are lower in Table 4 because OOVs lost during decoding miss post-decoding

Method	OOV Types w/ atleast 1 Candidate Generated		OOV Types w/ Correct Candidates Detected		OOV Tokens w/ Correct Candidates Detected	
Total OOVs Present	1975	-	1975	-	2970	-
All 3 Methods Combined	1974	99.9%	989	50.1%	1564	52.6%
Levenshtein Distance	1651	83.6%	475	24.1%	711	23.9%
Word Embedding	1104	55.9%	169	8.6%	236	7.9%
Transliteration	1886	95.5%	635	32.2%	1060	35.7%

Table 4: Candidate translations matched in reference sentences (Hindi)

Method	OOV Types w/ atleast 1 Candidate Generated		OOV Types w/ Correct Candidates Detected		OOV Tokens w/ Correct Candidates Detected	
Total OOVs Present	3719	-	3719	-	4846	-
All 3 Methods Combined	3719	100%	1139	30.6%	1453	30.0%
Levenshtein Distance	2841	76.4%	696	18.7%	855	17.6%
Word Embedding	1596	42.9%	215	5.8%	263	5.4%
Transliteration	3719	100%	395	10.6%	551	11.4%

Table 5: Candidate translations matched in reference sentences (Uzbek)

methods in the second column of Table 4. The middle two columns show that on searching in the reference sentence, we match candidates for 50% of OOV types in Hindi. The last two columns present these statistics in terms of OOV tokens. Method-specific OOV coverage is presented in the last three rows while the second row shows the overall coverage computed through a union of these methods.

Let us now turn to Uzbek. Table 5 lists the number of OOV types correctly matched in the reference sentence by running our pipeline on Uzbek data. As noted in Section 4.1, the Uzbek training data is much smaller than that of Hindi and has many more OOVs. Correspondingly, the upper bound achieved in Uzbek is lower.

4.4 Results

Here, we present the results obtained by integrating the translation candidates using the methods discussed in Section 3.4.

Classifier: To perform the classification experiments, we hand-tagged ≈ 1200 Hindi OOV words into the categories discussed in Section 3.1. The distribution is shown in Table 6. We used the Hindi POS Tagger by Reddy and Sharoff (2011) for generating features.

The best accuracy obtained with these features was 35.9%.[8] This was too low to be used to make decisions about which translation method to use. In addition, several other drawbacks were identified in this approach. Firstly, gold-standard labels are obtained through manual annotation, which requires time, money, and knowledge of the language. Secondly, the distribution of OOV words across these categories is uneven, making it difficult to achieve a high accuracy for smaller classes with limited data. Lastly, for training the classifier, the part of speech tags (POS) of the OOV and its context words are important features. However, low resource languages do not always have such tools readily available.

XML Markup and Secondary Phrase Table: We use Moses as the Statistical Machine Translation (SMT) system to run our translation experiments. Settings for the Hindi baseline system have been derived from Edinburgh's submission for WMT-2014 (Durrani et al., 2014a) because the same dataset is being used here. We use basic Moses settings for Uzbek.

transliteration

[8]Note that these are preliminary results and were not explored further due to the drawbacks identified

Category	Percentage
Source Content Words	22.2%
Named Entities	35.5%
Borrowed Words	28.7%
Misspellings & Typos	7.4%
Acronyms	3.2%
Numbers & Punctuation	1.0%
Transliterated English Words with Hindi Inflection	1.9%

Table 6: Distribution of OOV categories

Method	BLEU Score	OOV Types Detected as Correct
Baseline	12.15	57[10]
Transliteration	12.49	593
Transliteration + Bigger LM	12.67	616[11]
XML Markup	12.61	412
Secondary Phrase Table	12.16	372
Secondary Phrase Table without multi-word candidates	12.30	376
Secondary Phrase Table with frequency as feature	12.02	281
Oracle	13.48	989

Table 7: Results of integration of OOV candidates with SMT pipeline (Hindi)

Table 7 shows results of running Moses with the various integration options. We use BLEU as the evaluation metric (Papineni et al., 2002). We run experiments with the baseline system with no OOV translation, i.e. all OOVs in Hindi script are copied as is in the translated English output. We also run only-transliteration experiments with both the original and bigger language model. Since transliteration is integrated in Moses and we generate only one-best transliteration, these experiments do not require any system to prune or pick a translation. Using this method alone to translate all OOVs is good for languages that are related or have many named entities. We also present results of using XML Markup and secondary phrase table for integration of new translation pairs. In addition, results for the two proposed variations of secondary phrase table are included. First, we add inverse of frequency of the English candidate as an additional feature to the phrase table. Frequency for English words is computed using a large English corpus of past WMT news data.[9] Second, we discard all multi-word synonyms obtained from WordNet. This is because we address single word OOVs here and they are more likely to translate into single word candidates.

In addition to BLEU score, we also record the number of correct OOV translations in the final Moses output. Once we know the correct translation for an OOV by searching for the candidates in the reference sentence, we check if this correct translation appears anywhere in the translated output sentence. In other words, out of the OOVs which obtain a correct translation among their translation options, how many of them had the correct one picked through the SMT system. We use our technique of searching for the translation *anywhere* in the sentence. The possibility of false positives noted in the method of measuring oracle performance also exists in this analysis. But, as mentioned before, there are only a few of those.

Using only transliteration for Hindi-English gives the best performance in terms of both BLEU scores and number of OOVs translated correctly. One of the reasons for this is the high

[9] `http://www.statmt.org/wmt12/`

[10] These are either numerical or other OOVs, like words in English script, that are correctly translated when copied

[11] This number differs from the higher oracle performance of transliteration (635) in Table 4 because synonyms are also included in our system's transliteration candidates

Method	BLEU Score	OOV Types Detected as Correct
Baseline	9.93	394[10]
XML Markup	8.91	455
Secondary Phrase Table	9.77	507
Oracle	10.18	1139

Table 8: Results of integration of OOV candidates with SMT pipeline (Uzbek)

percentage of English words that tend to be used in Hindi language, as shows in Table 6. Furthermore, being news data, there are many named entities. According to our small hand-tagged set in Table 6, these two classes together constitute about 65% of the OOVs. The current integration methods lack sophisticated feature sets that can pick the correct candidate for an OOV from among its 40 (on an average) generated candidates and their synonyms. Discarding multi-word candidates as a basic pruning technique only shows slight improvements. The decrease in performance on adding inverse frequency of candidates as a feature is likely to be due to strong bias for low frequency words. Not all OOVs are rare words, especially in low resource settings, and strongly favoring very unfamiliar translation candidates can lead to a worse performance, unless combined with other useful features.

We also present results of integration of OOV candidates in the SMT pipeline on Uzbek-English data in Table 8. Since Uzbek uses Roman script, simply copying the OOVs in to the output helps translate some OOVs, which is why the baseline and transliteration method are the same.

These results show that we obtain comparable BLEU scores across all the techniques. On re-running identical experiment setups, it was observed that the BLEU scores varies by ± 0.35. Furthermore, an increase in number of OOVs correctly translated did not *always* result in a corresponding higher BLEU score. This suggests that in addition to the augmented OOV translations, other factors also have a considerable influence on the BLEU score, making it less reliable for this task. Also, while we obtain good quality candidates for OOV as seen in the oracle performance, the pipeline for integration requires additional features and pruning techniques to be able to pick the one correct candidate.

5 Conclusion and Future Work

We present a system that applies a combination of language-independent techniques to translate OOV words in the absence of large amounts of parallel data, a significant problem for low resource languages. In addition, we also present ways to evaluate these techniques and integrate the generated candidates into an SMT pipeline.

Here, we discuss the current limitations and possible future improvements for each translation method proposed:

- **Word Embeddings:** As we make more progress in word embeddings, we can understand the dimensions and limit it from producing antonyms and distant words. In addition, there are known methods like Deep CCA, which find *non-linear* projections for vectors in the two views (Andrew et al., 2013; Lu et al., 2015). We do not include that in the scope of this work because they require extensive tuning of parameters. For languages with related rich-resource languages, Generalized CCA has the ability to leverage more than two views of data to find a shared subspace with richer and more informed embeddings as has been shown in previous work (Rastogi et al., 2015). Furthermore, to make the cosine distance-based search faster, one can use approximation techniques (Zhao et al., 2015) like Locality Sensitive Hashing (Gionis et al., 1999) and Redundant Bit Vectors (Goldstein et al., 2005), which could not be included in the scope of current work.

- **Levenshtein Distance:** Although Levenshtein distance-based candidates show exact match for a good percentage of OOV words, they generate a lot of candidates. With no score other than the distance from OOV word, there is no way to rank these candidates. In a more resource-rich setting, features such as part of speech tag can be used to prune the list by removing words that do not have a matching tag. When working with a specific language which has, for instance, seen morphological variants for a majority of its OOVs, one can limit to a suffix or prefix based Levenshtein distance to keep the candidate list short.

- **Transliteration:** In this work, we only produce the 1-best transliteration using Moses. But, with better ways of ranking translation options, one can generate an n-best list of transliterations using the same module to widen the scope of coverage for named entities and rectify spelling errors.

To improve the end-to-end performance of this system, better ways of scoring and picking from among the candidate translations are required in addition to the above translation methods. For instance, for domains like news data with a high percentage of named entities, one can use named entity recognition to classify OOVs. The limitations imposed by working in a language-independent and low resource setting make implementing such methods less straightforward. Rich features cannot be created due to low amount of data as well as limited language-specific tools. A seperate phrase table could be used for each method, allowing for method-specific weights and manual tuning.

In conclusion, we present a completely unsupervised pipeline that applies a combination of techniques to translate OOVs and integrates them with an SMT system. We also propose ways to evaluate and measure upper bounds on the performance of these techniques. While there is a scope of improvement in how these candidates are integrated and picked by the SMT system, we obtain a good potential improvement in finding translations for unknown words in low-resource settings.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-15-C-0113. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). We thank Rebecca Knowles for useful discussions and suggestions. We thank the reviewers for their comments and suggestions.

References

Al-Rfou, R., Perozzi, B., and Skiena, S. (2013). Polyglot: Distributed word representations for multilingual NLP. *CoNLL-2013*, page 183.

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1247–1255.

Banerjee, P., Naskar, S., Roturier, J., Way, A., and van Genabith, J. (2012). Domain adaptation in smt of user-generated forum content guided by oov word reduction: Normalization and/or supplementary data. In *Proceedings of the 16th Annual Meeting of the European Association for Machine Translation, Trento, Italy*, pages 169–176.

Bird, S. (2006). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Bojar, O., Diatka, V., Rychlý, P., Straňák, P., Suchomel, V., Tamchyna, A., and Zeman, D. (2014). HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Buck, C., Heafield, K., and van Ooyen, B. (2014). N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavk, Icelandik, Iceland.

Daumé III, H. and Jagarlamudi, J. (2011). Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 407–412. Association for Computational Linguistics.

Durrani, N., Haddow, B., Koehn, P., and Heafield, K. (2014a). Edinburgh's phrase-based machine translation systems for WMT-14. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation, Baltimore, MD, USA*, pages 97–104.

Durrani, N., Sajjad, H., Hoang, H., and Koehn, P. (2014b). Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*, pages 148–153.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529.

Goldstein, J., Plat, J. C., and Burges, C. J. (2005). Redundant bit vectors for quickly searching high-dimensional regions. In *Deterministic and Statistical Methods in Machine Learning*, pages 137–158. Springer.

Habash, N. (2008). Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 57–60. Association for Computational Linguistics.

Habash, N. and Metsky, H. (2008). Automatic learning of morphological variations for handling out-of-vocabulary terms in Urdu-English machine translation. *Proceedings of the Association for Machine Translation in the Americas (AMTA-08), Waikiki, HI*.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Irvine, A. and Callison-Burch, C. (2013). Supervised bilingual lexicon induction with multiple monolingual signals. In *HLT-NAACL*, pages 518–523.

Koehn, P. and Haddow, B. (2009). Edinburgh's submission to all tracks of the WMT2009 shared task with reordering and speed improvements to moses. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 160–164. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Lu, A., Wang, W., Bansal, M., Gimpel, K., and Livescu, K. (2015). Deep multilingual correlation for improved word embeddings. In *Proceedings of NAACL*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., Yih, W., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Rastogi, P., Van Durme, B., and Arora, R. (2015). Multiview LSA: Representation learning via generalized CCA. In *Proceedings of NAACL*.

Reddy, S. and Sharoff, S. (2011). Cross language POS taggers (and other tools) for indian languages: An experiment with Kannada using Telugu resources. In *Proceedings of the Fifth International Workshop On Cross Lingual Information Access*, pages 11–19, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Řehůřek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. `http://is.muni.cz/publication/884893/en`.

Zhang, J. and Zong, C. (2013). Learning a phrase-based translation model from monolingual data with application to domain adaptation. In *ACL (1)*, pages 1425–1434.

Zhao, K., Hassan, H., and Auli, M. (2015). Learning translation models from monolingual continuous representations. In *Proc. NAACL*.