UEdin participation in the 1st Translation Memory Cleaning Shared Task

Citation for published version:
Buck, C & Koehn, P 2016, UEdin participation in the 1st Translation Memory Cleaning Shared Task. in Proceedings of 2nd Workshop on Natural Language Processing for Translation Memories (NLP4TM 2016). Portorož, Slovenia, 2nd Workshop on Natural Language Processing for Translation Memories, Portorož, Slovenia, 28/05/16.

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of 2nd Workshop on Natural Language Processing for Translation Memories (NLP4TM 2016)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
UEdin participation in the 1st Translation Memory Cleaning Shared Task

Christian Buck, Philipp Koehn
University of Edinburgh, Johns Hopkins University
Scotland, Baltimore MD
christian.buck@ed.ac.uk, phi@jhu.edu

Abstract
We present our submission for the 1st Translation Memory Cleaning Shared Task. We treat the task as a 3-class classification problem and extract features that indicate (i) source sentence complexity, (ii) misalignments between source and target, and (iii) target sentence complexity. Our results show that focusing on the target side and finding ways to estimate the alignment quality between source and target yields expressive features which, together with a reliable classifier, produces competitive results. Our submission is ranked on 2nd place among 6 for the EN-DE language pair.

Keywords: translation memory cleaning, sub-word unit language model, neural sentence alignment

1. Introduction
Parallel corpora are the most relevant resource for building translation tools such as Machine Translation (MT) models, bilingual concordancers, and interactive post-editing environments. As all natural language resources parallel data is subject to a certain level of noise originating for either problems in the data collection process, such as faulty sentence-alignments, or simply incorrect or incomplete translations produced by translators. While some applications such as Statistical MT systems can still benefit from data with low to medium levels noise (Smith et al., 2013), applications where the incorrect translation is visible to the user are more sensitive.

Translation Memories (TMs) are commonly used as a source for target side suggestions in a post-editing setting. A segment pair is selected based on a similarity score between the source segment that is to be translated and the source side of the TM pair. In this setting low quality sentence pairs can have a negative impact on productivity and should be filtered beforehand.

2. Shared Task
For the 1st Translation Memory Cleaning Shared Task annotated datasets in three language pairs – English–Spanish, English–Italian, and English–German – are provided. In this work we only deal with the last pair. However, our methods are language-independent and could be used for other language pairs.

For English–German, the organizers provide 1396 training pairs and 700 test pairs, the former annotated with a 3-class target variable indicating quality. Table 1 outlines the annotation guidelines. In addition, the organizers propose two binary classification tasks where a translation is wrong if it (Binary I) belongs in either class (2) or (3), or (Binary II) belongs in class (3).

As evident from Table 2 the data exhibits significant class imbalance with most examples being correct translations.

3. Preprocessing
We tokenize all text using the subword method introduced by Sennrich et al. (2015). This method is aimed at reducing the vocabulary size by splitting words into smaller units based on a dictionary of known character n-grams. The dictionary is populated by incrementally replacing the most common character bigram in a corpus with a new character, which may in turn become part of another new dictionary entry. Thus every new character represents 2 or more characters of the original word. Text is now tokenized by selecting the fewest splits for each word such that every part occurs in the dictionary. In the example below | marks a token boundary:

original: a Primer for Pandemics
word units: a Pr | im | er for Pa | nd | em | ic | s

In our experiments we use a dictionary of size 50k which was estimated on a corpus of 4.2M sentences from the Europarl and News datasets distributed by the WMT15 evalu-
We train a 5-gram language model using the same corpus. As shown in Figure 1 after accounting for sentence length, a low log-probability is a good indicator of incorrect sentence pairs.

4. Features
In this work we focus mostly on extracting expressive features and use an off-the-shelf machine learning approach to combine these. We use a number of features that have been successfully used in Quality Estimation (Buck, 2012; De Souza et al., 2013a; De Souza et al., 2014). We group these into five categories:

Surface features (12 features)

- Number of source tokens after tokenization
- Number of target tokens after tokenization
- Number of characters in source/target and their ratio
- Number of tokens classified as number in source/target and their ratio
- Number of non-alphanumeric tokens in source/target and their ratio
- Binary indicator telling if one side ends with non-alphanumeric character and the other one doesn’t

Language Model (LM) (8 features, 4 per language)
For our language model features we include both the probability $P(w_i^N)$ of a sentence $w_i^N$ and the perplexity $PP(w_i^N) = P(w_i^N)^{-\frac{1}{N}}$. We use logarithmic values for these.

- LM (log-)probability/perplexity of source/target on a 5-gram Kneser-Ney smoothed language model that was estimated on Europarl and NewsCrawl corpora using the subword units mentioned in Section 3.
- LM (log-)probability/perplexity of source/target on very large language models trained on CommonCrawl data as described in Buck et al. (2014). The 5.5 TB English model is estimated on roughly a billion words.

Word Alignment (5 features)
Following the work of De Souza et al. (2013b), we compute a number of features based on word alignments. For these features we use the Moses (Koehn et al., 2007) tokenizer.

- Model score of fast_align (Dyer et al., 2013) model trained on WMT16 EN/DE training data consisting of 4.6M lines. Unfortunately, due to a software bug, this score was unreliable and the feature is therefore not part of our submission. Both directions, i.e. EN-DE and DE-EN, are aligned.
- Number of unaligned source/target words based on symmetrized alignment

Figure 1: Target length vs. language model log-probability on EN-DE training data.
Neural Alignment Score (4 features)

Following [Buck (2012)], we use a feed-forward neural network to estimate the relation between source and target. To encode a sentence we use a bloom-filter as a fixed-length representation of a bag-of-bigrams. For each sentence, we extract all bigrams and then use several hash functions to populate fields in binary vector of fixed length. In our experiments we use either 1024 or 2048 dimensions and 5 hash functions. In contrast to previous work [Buck, 2012] recent advances in neural network algorithms and implementations allow us to quickly train the network with several hidden layers. We use two hidden layers with half the number of nodes in the input layer. We train the model to optimize cross-entropy loss:

\[ L(X, Y) = -\frac{1}{N} \sum_{n=1}^{N} [y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n)] \]

where \( X \) and \( Y = y_1^N \) are the binary input and (expected) output vectors and \( \hat{y}_n \) is the activation of the \( n \)th output neuron. On the output layer we use sigmoid activations and rectified linear units otherwise. We use Keras[https://github.com/Theano/Theano] for GPU training and Adam [Kingma and Ba, 2014] as the optimizer.

Our model is trained on a small sample of 200k lines of parallel WMT16 EN-DE news data. The training takes about 30min to 1h on a (shared) NVIDIA Titan X GPU. Once the model is trained it can be evaluated very quickly, even on a CPU.

While we optimize for cross-entropy when training the network, we found that the local loss does not produce a very predictive feature. Instead we use both cosine similarity

\[ \text{sim}_\text{cos}(\hat{y}_1^N, \hat{y}_1^N) = \frac{\sum_{n=1}^{N} y_n \hat{y}_n}{\sqrt{\sum_{n=1}^{N} y_n^2} \sqrt{\sum_{i=1}^{N} \hat{y}_i^2}} \]

and the maximum entry in \( \hat{y} \) that should be zero:

\[ \text{sim}_\text{maxneg}(\hat{y}_1^N, \hat{y}_1^N) = \max_{n=1}^{N} \{(1 - y_n) \hat{y}_n\} \]

as features and stop training based on the Pearson correlation coefficient between these features and the training classes.

Table 3: Results on the test set. For the 3-class case reported numbers are average \( F_1 \) weighted by label frequency, for the binary tasks reported numbers are unweighted average \( F_1 \). The rightmost column shows the number of correctly classified instances in the 3-class setting, out of a total of 700. Post submission results are not part of the shared task but are given to illustrate the performance difference between full and selected feature sets as described in Section 7.

5. Experiments

We treat the task as a 3-class classification problem and derive predictions for the binary cases from the finer grained predictions. As already evident from Figure 1 distinguishing between classes (2) and (3) is challenging, whereas many correct pairs seem to be easy to identify.

To select a predictive model and hyper parameters we use 10-fold stratified cross-validation with fixed folds on the training data. We performed experiments with common ML techniques including SVMs, Neural Networks, Maximum Entropy models, and KNN to varying degrees of success. In general, we found that performance varied heavily between cross-validation folds, possibly because of brittle hyper parameters. Furthermore, some of the aforementioned models require pre-processing of the feature set such as scaling/standardization and removal of outliers. We found RandomForests [Breiman, 2001] to give reliable and competitive performance across folds, without the need to select many hyper parameters and feature transformations and use them in all experiments below.

6. Results

For the shared task we submit two systems that differ only by a single feature, the Neural MT model score. The motivation for this is to show to what extent adding this computationally expensive feature improves performance. As shown in Table 3 both feature sets lead to similarly well-performing models. The table also reports results on two baseline systems provided by the organizers, one producing random assignments and one based on sentence lengths. Our system clearly outperforms these baselines and is ranked 2nd among 6 participants.

7. Feature Selection

To see which features are the most important we perform recursive feature elimination based on cross-validation on the training data. In each step we remove one feature, until
Table 4: Features selected using recursive feature elimination. The ranking is based on the feature importance assigned by the Random Forest classifier.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subword LM perplexity target</td>
</tr>
<tr>
<td>2</td>
<td>Subword LM probability target</td>
</tr>
<tr>
<td>3</td>
<td>CommonCrawl LM perplexity target</td>
</tr>
<tr>
<td>4</td>
<td>Neural Alignment MaxNeg 2048dim</td>
</tr>
<tr>
<td>5</td>
<td>Number of unaligned source words</td>
</tr>
<tr>
<td>6</td>
<td>Neural Alignment MaxNeg 1024dim</td>
</tr>
<tr>
<td>7</td>
<td>Number of unaligned target words</td>
</tr>
<tr>
<td>8</td>
<td>CommonCrawl LM probability target</td>
</tr>
<tr>
<td>9</td>
<td>Word alignment model score</td>
</tr>
<tr>
<td>10</td>
<td>Number of characters in target</td>
</tr>
<tr>
<td>11</td>
<td>Number of subword units in target</td>
</tr>
</tbody>
</table>

performance deteriorates. Table 4 shows that most relevant features are based only on the target sentence. Among those language models seem to be the most indicative. The two top-ranking LM features are based on subword units which were originally devised (Sennrich et al., 2015) to overcome vocabulary size limitations in neural machine translation. Besides the LM features we find two features based on the bloomfilter-to-bloomfilter neural alignment, along with other features based on word alignments. All remaining relevant features are based on the length.

We report post-submission results in Table 5. These are slightly improved over our submission results due to (i) fixed error in word alignment scores (ii) fixed error in length normalization for language models. Reducing the number of features to 11 results in slightly lower performance. A possible reason is that the repeated use of cross-validation on the training has led to slight overfitting.

8. Conclusion

We presented our submission for the 1st Translation Memory Cleaning Shared Task. A number of features ranging from shallow to computationally expensive are produced and used in conjunction with a Random Forest classifier to detect incorrect translations. We find that features based on target side language models, word alignment, and a neural alignment model are the most discriminative and yield competitive performance.

9. Acknowledgements

We would like to thank Rico Sennrich and Matthias Huck for helpful discussions and aid with running the MT systems.


