Ensembles of Classifiers for Cleaning Web Parallel Corpora and Translation Memories

Eduard Barbu
Institute Of Computer Science / University of Tartu
eduard.barbu@ut.ee

Abstract

The last years witnessed an increasing interest in the automatic methods for spotting false translation units in translation memories. This problem presents a great interest to industry as there are many translation memories that contain errors. A closely related line of research deals with identifying sentences that do not align in the parallel corpora mined from the web. The task of spotting false translations is modeled as a binary classification problem. It is known that in certain conditions the ensembles of classifiers improve over the performance of the individual members. In this paper we benchmark the most popular ensemble of classifiers: Majority Voting, Bagging, Stacking and Ada Boost at the task of spotting false translation units for translation memories and parallel web corpora. We want to know if for this specific problem any ensemble technique improves the performance of the individual classifiers and if there is a difference between the data in translation memories and parallel web corpora with respect to this task.

1 Introduction

Translation memories are databases that store sentence segments in one language with their corresponding translation in another language. The origin of translation units in translation memories is diverse. Mostly, they come from professional translators, but some translation memory databases (e.g. MyMemory (Trombetti, 2009)) accept contributions from online contributors. Translation memories are integrated in computer-assisted translation tools and they are among the most used repositories of information by professional translators. For various reasons (e.g. the negligence of the translators, malicious users that spam through collaborative interfaces), a significant number of translation units in translation memories are false translations.

However, a similar resource can be obtained by mining the web (or large crawled documents repositories like Common Crawl) for parallel corpora. The pipeline for web mining starts with parallel sites discovery, parallel web documents identification and finishes with the alignment of parallel sentences in the documents. The resulting resource contains errors. This happens because either the automatic document pairing fails or because the sentence aligning fails.

Below there is an example of two translation units for the language pairs English-Italian and English-French that should be discarded.

1. An example for English-Italian language pair extracted from a translation memory.
   English segment. Weight loss category.
   Italian segment. Studi sulle linee di categoria.
   English translation of the Italian segment : “Studies concerning categories”.

2. An example for English-French language pair produced by mining the web.
   English segment. The hotel has a minivan shuttle and took us on a complementary basis to several places which were a bit far to walk as one of us was injured skiing.
   French segment. Nous allons vivre un moment magique...aperitif dans la “bodega” , puis menu degustation dans une petite salle intimiste.
   English translation of the French segment: “We will live a magic moment ... aperitif in

Proceedings of Recent Advances in Natural Language Processing, pages 71–77,
Varna, Bulgaria, Sep 4–6 2017.
https://doi.org/10.26615/978-954-452-049-6_011

the "bodega", then menu tasting in a small intimate room."

The pace at which translation memories grow nowadays and the web parallel corpora are built makes it impossible to perform manual cleaning[1]. Given the size of modern translation memories and parallel corpora, automatic solutions have been proposed. They state the problem as a binary classification task. Given a translation unit, a classifier should return "yes" if the segments in the translation unit are true translation and "no" otherwise.

In this paper we explore various methods for building ensembles of classifiers to tackle the problem of spotting false translations in translation memories and web parallel corpora. It has been shown (Dietterich, 2000) that the ensembles of classifiers can work better than the individual classifiers in certain conditions. We want to know what the difference is between the data coming from web parallel corpora and the data coming from translation memories with respect to this task.

The rest of the paper has the following organization. The next section presents the related work. In section 3, the features used and the ensemble of classifiers are introduced. In section 4 we describe the data used for training and testing the classifiers and the results of the classifiers evaluation. The paper ends with the conclusions.

2 Related Work

Most people that mine large corpora from the web use Gale-Church score (Gale and Church, 1993) to filter the translation units. Please notice that Moses statistical machine translation engine includes a python script[2] for this purpose. However, this simple technique is clearly insufficient as none of the translation units in the examples above can be discarded with the Gale-Church score. According to our knowledge, the first study that uses a classifier to identify sentences that are translations in (comparable) corpora is Munteanu (2005). They use a maximum likelihood classifier trained on a set of features that are similar to the word alignment features we use (see section 3 for details) .

[1]For example, MyMemory receives approximately 15 million contributions per month.

[2]The script is called gacha.py

In Barbu (2015) we were the first to train a set of supervised classifiers to spot false translations in translation memories. An unsupervised approach for the same task was proposed in Jalili Sabet (2016). They use a set of features that capture the similarity between the segments in a translation unit, use them to induce training labels and train a set of base classifiers with Extremely Randomized Trees (Geurts et al., 2006). The most relevant attempts to spot false translation units in translation memories are surveyed in Barbu (2016). One of the most successful systems is trained not only on features related to translation quality, but also on features related to grammatical errors and features related to fluency and lexical choice (Wolff, 2016).

The previous supervised works focused on tuning and evaluation of the individual classifiers. The only research that uses ensemble of classifiers are Barbu (2015) which trained a Random Forest and Jalili Sabet (2016) which used Extremely Randomized trees. However, they did not attempt to combine individual classifiers in a systematic way and compare the performance of the individual classifiers with the performance of the ensembles of classifiers. Moreover, we are the first to test the classifiers on data coming from both parallel corpora and translation memories in a multilingual setting. This is important because there are data providers that want to clean the translation memories and the data collected from the web using a single trained model per language pair.

3 Classification

In this section we first present the individual classifiers and the motivation for choosing them, then the ensemble of classifiers and finally the features computed for the translation units to be classified.

3.1 Classifiers

The individual classifiers selected are among the widely used classifiers in the literature : *Decision Tree*, *Logistic Regression*, *Support Vector Machines* with the linear kernel, *Support Vector Machines* with the radial basis function kernel, *K-Nearst Neighbors*.

The ensembles of classifiers are meta-classifiers that combine the results of multiple classifiers. We presume that the ensemble of classifiers has a better generalization performance than the individual classifiers. We have explored the following popu-

lar ensemble techniques.

- *Majority Voting*. The meta-classifiers' predicted class is the class mostly voted by the individual classifiers. For example, if we have an ensemble composed of three classifiers and the vote assigned to a translation unit is (1,1,0), that is the first two classifiers predict that the result is positive and the third classifier that the result is negative, then the meta-classifier chooses the positive class. It can be proven that if the classifiers are independent and the error rated are not correlated, then the majority voting error rate is lower than the individual classifiers error rate (Dietterich, 2000).

- *Stacking*. As in majority voting case the individual classifier's predictions are combined. However, the final decision is taken by another classifier that is trained on the labels outputted by the classifiers in the ensemble.

- *Ada Boost*. Boosting is a machine learning technique that creates an accurate prediction combining many weak classifiers. A weak classifier is a classifier that performs slightly better than random guessing. Each weak classifier is trained on a random set of the training set. Ada Boost (Freund and Schapire, 1997) assigns weights to each training example. The assigned weight represents the probability that the training example appears in the training set. At each step the examples that were incorrectly predicted by a classifier have their weights increased and the examples that were correctly predicted have the weights decreased. Ada Boost assigns weights to each classifier based on the classifier accuracy. The classifiers that have better accuracy receive higher weight.

- *Bagging* Bagging (bootstrap aggregating) (Breiman, 1996) is an ensemble technique where the individual classifiers in the ensemble are trained on bootstrap samples (random samples taking from initial training set with replacement). The prediction for the test set is done as in the *Majority Voting* ensemble.

3.2 Features

The features used by the classifiers and computed for both training and test sets are either word alignment features, presence/absence features or miscellaneous features. The features are for the most part the features we have introduced before (Barbu, 2015). We have re-engineered some of them and replaced the similarity between the target segment and translation of the source segment with the word alignment features. In this way we no longer rely on an expensive machine translation system. Please notice that to compute some features, like word alignment features for example, you need a word aligner trained on a large parallel corpus.

1. Word Alignment Features. The idea behind word alignment features is that the word alignments provide an important clue for the hypothesis that source and target segments are translations. We compute the number of aligned words in source and target segments and the longest contiguous zone of aligned and unaligned words.

   - *Source (Target) Word Ratio*. The number of words in the source (target) segment that are aligned divided by the total number of words in the source (target) segment.
   - *Max Source (Target) Aligned Zone Ratio*. The number of words in the source (target) longest aligned continuous zone divided by the total number of words in the source (target) segment.
   - *Max Source (Target) Unaligned Zone Ratio*. The number of words in the source (target) longest unaligned continuous zone divided by the total number of words in the source (target) segment.

2. Presence/Absence Features. These features signal the presence/absence of an entity (URL, email address, named entity, etc.) in source and target segments. Furthermore, if the entities are present in both source and target segments, their ratio is computed. These features capture the intuition that if an entity is present in the source segment and if the target segment is a translation of the source segment it is very probable that the same entity is present in the target segment.

   - *Entity Features*. These features are *tag*, *URL*, *email*, *name entity*, *punctuation*, *number*, *capital letters*, *words in capital*

letters. The value of these features is 1 if the source or target segments contain a tag, URL, email, name entity, punctuation, capital letters or words written in capital letters, otherwise is 0.

- *Entity Similarity Features*. If a feature explained under *Entity Features* except for *capital letters* and *words in capital letters* exists, the cosine similarity between the source and target segments entity vectors is computed. Therefore, we compute a feature for the tag similarity, punctuation similarity, URL similarity, etc.
- *Capital letters word difference*. The value of this feature is the ratio between the difference of the number of words containing at least a capital letter in the source segment and the target segment and the sum of the capital letter words in the translation unit. It is complementary to the feature *capital letters*.
- *Only capital letters difference*. The value of the feature is the ratio between the difference of the number of words containing only capital letters in the source segment and the target segments and the sum of only the capital letter words in the translation unit. It is complementary to the feature *words in capital letters*.

3. Miscellaneous Features. The rest of the features we compute fall under miscellaneous features category because they are not easily categorized.

   - *language difference*. If the language codes identified by a language detector for the source and target segments coincide with the language codes declared for the same source and target segments, then the feature is 1, otherwise is 0.
   - *Gale Church score*. This feature is the slightly Gale-Church score described in the equation 1 and introduced in Tiedemann (2011). This score reflects the idea that the length of the source (l_s) and target segments (l_d) that are true translations is correlated. We expect that the classifiers learn the threshold that separates the positive and negative examples. However, relying exclusively on the Gale-Church score is tricky because there are cases when a high Gale-Church score is perfectly legitimate. For example, when the acronyms in the source language are expanded in the target language.

$$CG = \frac{l_s - l_d}{\sqrt{3.4(l_s + l_d)}} \quad (1)$$

4 Results and Discussion

4.1 Data

The setting we presuppose is that of a data provider that has a huge amount of translation memories and data crawled from the web and wants to train a unique model per language pair to clean both translation memories and data collected from the web. As we showed in (Barbu, 2015) a cross validation setting tends to overestimate the performance of the model. Therefore the training and test data for the English-Italian language pair is a random sample of multiple translation memories inside the MyMemory database. The test data is enriched with translation units from an aligned parallel English-Italian web site. The training and test data for English-French is a random sample from approximately 17 million translation units resulted from crawling and aligning at the sentence level a large number of parallel English-French sites. The crawling and alignment of the web sites have been performed with the ILSP crawler (Papavassiliou et al., 2013). The training and test sets have been annotated with positive and negative labels by an annotator and checked by a supervisor. The size of the training and test data and the distribution of positive and negative examples is given in table 1. The percentage of negative data in training and test set for English-French language pair is around 50%. This percentage is much higher than the percentage of negative data in English-Italian data sets. These figures are not a surprise because the data in translation memories has less noise than the data crawled and aligned from the web.

The features are computing using TM Cleaner, a publicly available software written by the author [3] . The individual classifiers and almost all ensembles of classifiers used in the experiments are im-

[3]https://github.com/SoimulPatriei/TMCleaner/tree/master/TMCleaner-MMT-API

Set	Training		Test	
Language Pair	en-it	en-fr	en-it	en-fr
Positive	2764	510	356	144
Negative	901	436	83	151
Total	3665	946	439	295

Table 1: The distribution of positive and negative translation units in training and test sets

Classifier	**F1 Positive**	**F1 Negative**	**Balanced Accuracy**
RBF	0.93	0.72	0.82
KN	0.93	0.65	0.77
DT	0.93	0.69	0.78
SK	0.93	0.69	0.8
BG	0.93	0.69	0.81
AB	0.93	0.73	0.84
MV	0.94	0.72	0.82
LR	0.94	0.73	0.82
LN	0.95	0.76	0.84

Table 2: Classification Results for English-Italian

plemented in the machine learning library scikit-learn (Pedregosa et al., 2011).

The word alignment features are computed using the fast align (Dyer et al., 2013) version modified in Modern Machine Translation project[4]. This version allows training a word alignment model on a parallel corpus and the use of the aligned model for obtaining the word alignments on a test set. The fast align in Modern Machine Translation is trained on English-Italian and English-French parallel corpora containing approximately 100 millions parallel words each.

The language codes assigned to English, Italian and French segments are computed with the language detector Cybozu [5].

The *Majority Voting* and the *Stacking* ensembles combine the results of all individual classifiers presented in section 3. The *Stacking* meta-classifier is logistic regression. *Ada Boost* uses 500 decision trees weak learners. The *Bagging* ensemble technique combines the results of 500 decision trees classifiers.

The results for English-Italian and English-French are presented in tables 2 and 3.

For each individual and ensemble of classifiers

[4] http://www.modernmt.eu

[5] https://github.com/shuyo/language-detection

Classifier	**F1 Positive**	**F1 Negative**	**Balanced Accuracy**
KN	0.81	0.77	0.8
DT	0.84	0.83	0.83
MV	0.86	0.86	0.86
LR	0.86	0.85	0.85
LN	0.86	0.86	0.86
RBF	0.86	0.86	0.86
AB	0.86	0.86	0.86
SK	0.87	0.86	0.86
BG	0.88	0.88	0.88

Table 3: Classification Results for English-French

we compute the F1 score for positive and negative class and the balanced accuracy (BA). The first column lists a short name form for the classifiers. RBF stays for *Support Vector Machine* with RBF kernel, KN for *K-Nearst Neighbors*, DT for *Decision Tree*, SK for *Stacking*, BG for *Bagging*, AB for *Ada Boost*, MV for *Majority Voting*, LR for *Logistic Regression* and LN for *Support Vector Machines* with linear kernel. In both tables the classifiers are ordered by the F1 score for the positive class. The ensembles of classifiers are emphasized with bold font.

In figures 1 and 2 the F1 positive scores and F1 negative scores are plotted for English-Italian and English-French language pairs. With continuous line we connect the F1-scores for the positive class and with dashed line we connect the F1-scores for the negative class. As in the above tables the classifiers are ordered in ascending order according to F1 positive score.

For English-Italian language pair the best F1-scores for positive and negative classes are obtained by *Support Vector Machines* with linear kernel. For English-French language pair the clear winner is the Bagging ensemble. The difference between the performances of the classifiers for positive and negative classes is substantial for the English-Italian test set. The difference between the best performing classifiers for the positive class (*Support Vector Machines* with linear kernel) and negative class (*Support Vector Machines* with linear kernel) respectively is 19 percents. The difference between the worst performing classifier for the positive class, *Support Vector Machines* with RBF kernel and the worst performing classifier for the negative class *K-Nearst Neighbors* is 28 percents. Instead, for English-French test set, the

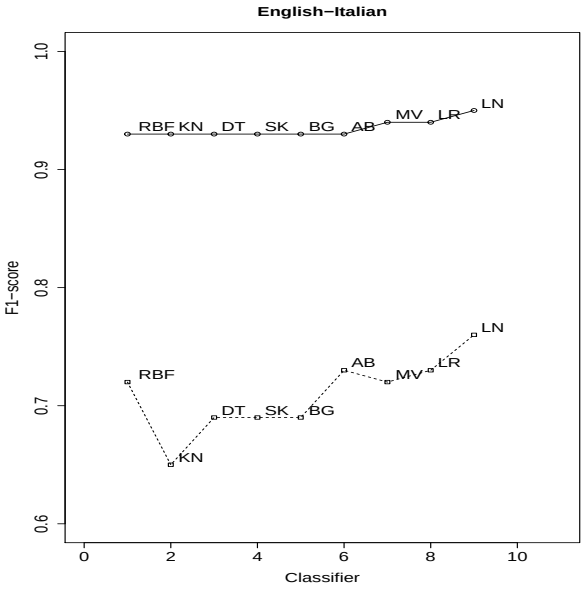

Figure 1: The F1-Scores for positive and negative class for English-Italian

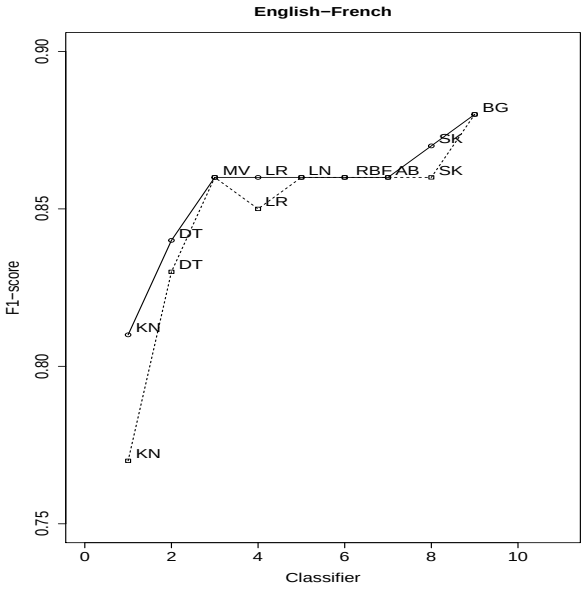

Figure 2: The F1-Scores for positive and negative class for English-French

difference between the best performing classifiers for the positive and negative classes is 4 percent. There is no difference between the worst performing classifiers for the positive and negative classes.

These results lead to the conclusion that it is much harder to distinguish between a positive and a negative translation unit in a translation memory than in parallel corpora crawled from the web. The translation units in parallel web corpora contain errors due to misalignment. They are easier to spot because it is less probable that the aligned segments have overlapping vocabulary.

According to balanced accuracy measure, the *Ada Boost* ensemble is as good as *Support Vector Machine* with the linear kernel for the mixed English-Italian test set. The *Bagging* Ensemble is the best classifier for English-French test set with the other ensembles and the *Support Vector Machine* with Linear kernel coming next.

5 Conclusions

In this paper we have tested well-known ensembles of classifiers (*Majority Voting*, *Stacking*, *AdaBoost* and *Bagging*) in the task of finding translation units that are not true translations in translation memories and parallel web corpora. Translation memories curators and researchers that build parallel corpora for machine translations can benefits from the results of this study.

To evaluate the performance of ensemble classifiers we have compared them against individual classifiers like *Support Vector Machines* and *K-Nearst Neighbors*. English-Italian translation units have better quality than the English-French translation units. For English-Italian language pair the ensemble classifiers are no better than the individual classifiers. If we order them according to the balanced accuracy measure, *Ada Boost* ensemble is as good as *Support Vector Machine* with the linear kernel. If we evaluate the classifiers according to F1-scores for positive and negative class then the *Support Vector Machine* with the linear kernel is the winner. For English-French language pair the *Bagging* Ensemble wins for both Balanced Accuracy and F1-scores.

The ensemble classifiers are more appropriate for data crawled and aligned from the web parallel sites. However, the quality of the content of the parallel sites together with the crawling and the alignment methodology play a big role in the decision to use or not an ensemble classifier. If the parallel sites are of good quality and the mapping methodology is very accurate, then it might well be the case that the resulting translation units have a quality close to translation memories. In this case using an ensemble classifier does not improve over a classifier like *Support Vector Machines*. If one is interested in mining a large amount of web sites with variable content quality, then using ensemble classifiers could the solution. The final

destination of the translation units matters as well. If they are used to improve a translation memory, then the quality matters and even a 2-percent improvement in the classifier performance is a success. If the translation units are used to train a SMT system, then a slightly cleaner resource does not have an impact over the BLEU score or the translation quality.

6 Reproducibility

To reproduce the results in this paper please go to the link bellow. The folder the link is pointing to is inside TM Cleaner package. Inside the folder you will find a README file. Follow the instructions to run the individual classifiers, the ensembles of classifiers the evaluation scripts and the R scripts for reproducing the graphs in this paper:
`https://github.com/SoimulPatriei/TMCleaner/tree/master/TMCleaner-MMT-API/Reproducibility`

References

Eduard Barbu. 2015. Spotting false translation segments in translation memories. In *Proceedings of the Workshop Natural Language Processing for Translation Memories*. Association for Computational Linguistics, Hissar, Bulgaria, pages 9–16. http://www.aclweb.org/anthology/W15-5202.

Eduard Barbu, Carla Parra Escartín, Luisa Bentivogli, Matteo Negri, Marco Turchi, Constantin Orasan, and Marcello Federico. 2016. The first automatic translation memory cleaning shared task. *Machine Translation* 30(3):145–166. https://doi.org/10.1007/s10590-016-9183-x.

Leo Breiman. 1996. Bagging predictors. *Mach. Learn.* 24(2):123–140. https://doi.org/10.1023/A:1018054314350.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*. Springer-Verlag, London, UK, UK, MCS '00, pages 1–15. http://dl.acm.org/citation.cfm?id=648054.743935.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *HLT-NAACL*. The Association for Computational Linguistics, pages 644–648.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55(1):119–139. https://doi.org/10.1006/jcss.1997.1504.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *COMPUTATIONAL LINGUISTICS* .

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Mach. Learn.* 63(1):3–42. https://doi.org/10.1007/s10994-006-6226-1.

Masoud Jalili Sabet, Matteo Negri, Marco Turchi, and Eduard Barbu. 2016. An unsupervised method for automatic translation memory cleaning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 287–292. http://anthology.aclweb.org/P16-2047.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Comput. Linguist.* 31(4):477–504. https://doi.org/10.1162/089120105775299168.

Vassilis Papavassiliou, Prokopis Prokopidis, and Gregor Thurmair. 2013. A modular open-source focused crawler for mining monolingual and bilingual corpora from the web. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*. Association for Computational Linguistics, Sofia, Bulgaria, pages 43–51. http://www.aclweb.org/anthology/W13-2506.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jörg Tiedemann. 2011. *Bitext Alignment*. Number 14 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA, USA. https://doi.org/10.2200/s00367ed1v01y201106hlt014.

Marco Trombetti. 2009. Creating the world's largest translation memory. http://www.mt-archive.info/MTS-2009-Trombetti-ppt.pdf.

Friedel Wolff. 2016. Combining off-the-shelf components to clean a translation memory. *Machine Translation* 30(3):167–181. https://doi.org/10.1007/s10590-016-9186-7.