Spotting false translation segments in translation memories

Eduard Barbu
Translated.net
`eduard@translated.net`

Abstract

The problem of spotting false translations in the bi-segments of translation memories can be thought of as a classification task. We test the accuracy of various machine learning algorithms to find segments that are not true translations. We show that the Church-Gale scores in two large bi-segment sets extracted from MyMemory can be used for finding positive and negative training examples for the machine learning algorithms. The performance of the winning classification algorithms, though high, is not yet sufficient for automatic cleaning of translations memories.

1 Introduction

MyMemory[1] (Trombetti, 2009) is the biggest translation memory in the world. It contains more than 1 billion bi-segments in approximately 6000 language pairs. MyMemory is built using three methods. The first method is to aggregate the memories contributed by translators. The second method is to use translation memories extracted from corpora, glossaries or data mined from the web. The current distribution of the automatically acquired translation memories is given in figure 1. Approximately 50% of the distribution is occupied by the DGT-TM (Steinberger et al., 2013), a translation memory built for 24 EU languages from aligned parallel corpora. The glossaries are represented by the Unified Medical Language System (UMLS) (Humphreys and Lindberg, 1993), a terminology released by the National Library of Medicine. The third method is to allow anonymous contributors to add source segments and their translations through a web interface.

The quality of the translations using the first method is high and the errors are relatively few.

[1]https://mymemory.translated.net/

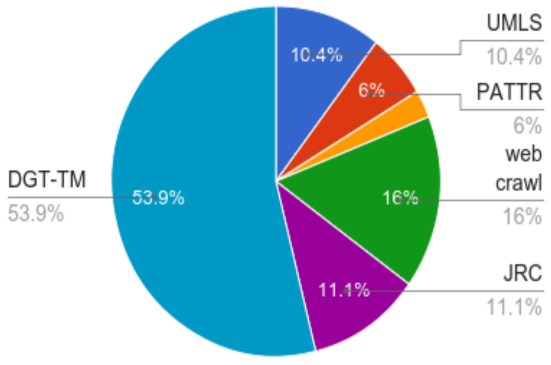

Figure 1: The distribution of automatically acquired memories in MyMemory

However the second method and especially the third one produce a significant number of erroneous translations. The automatically aligned parallel corpora have alignment errors and the collaborative translation memories are spammed or have low quality contributions.

The problem of finding bi-segments that are not true translations can be stated as a typical classification problem. Given a bi-segment a classifier should return yes if the segments are true translations and no otherwise. In this paper we test various classification algorithms at this task.

The rest of the paper has the following structure. Section 2 puts our work in the larger context of research focused on translation memories. Section 3 explains the typical errors that the translation memories which are part of MyMemory contain and show how we have built the training and test sets. Section 4 describes the features chosen to represent the data and briefly describes the classification algorithms employed. Section 5 presents and discusses the results. In the final section we draw the conclusions and plan the further developments.

2 Related Work

The translation memory systems are extensively used today. The main tasks they help accomplish are localization of digital information and translation (Reinke, 2013). Because translation memories are stored in databases the principal optimization from a technical point of view is the speed of retrieval.

There are two not technical requirements that the translation memories systems should fulfill that interest the research community: the accuracy of retrieval and the translation memory cleaning. If for improving the accuracy of retrieved segments there is a fair amount of work (e.g. (Zhechev and van Genabith, 2010), (Koehn and Senellart, 2010)) to the best of our knowledge the memory cleaning is a neglected research area. To be fair there are software tools that incorporate basic methods of data cleaning. We would like to mention Apsic X-Bench[2]. Apsic X-Bench implements a series of syntactic checks for the segments. It checks for example if the opened tag is closed, if a word is repeated or if a word is misspelled. It also integrates terminological dictionaries and verifies if the terms are translated accurately. The main assumptions behind these validations seem to be that the translation memories bi-segments contain accidental errors (e.g tags not closed) or that the translators sometimes use inaccurate terms that can be spotted with a bilingual terminology. These assumptions hold for translation memories produced by professional translators but not for collaborative memories and memories derived from parallel corpora.

A task somehow similar to translation memory cleaning as envisioned in section 1 is Quality Estimation in Machine Translation. Quality Estimation can also be modeled as a classification task where the goal is to distinguish between accurate and inaccurate translations (Li and Khudanpur, 2009). The difference is that the sentences whose quality should be estimated are produced by Machine Translations systems and not by humans. Therefore the features that help to discriminate between good and bad translations in this approach are different from those in ours.

[2]http://www.xbench.net

3 The data

In this section we describe the process of obtaining the data for training and testing the classifiers. The positive training examples are segments where the source segment is correctly translated by the target segment. The negative training examples are translation memory segments that are not true translations. Before explaining how we collected the examples it is useful to understand what kind of errors the translation memories part of MyMemory contain. They can be roughly classified in the four types :

1. **Random Text**. The Random Text errors are cases when one or both segments is/are a random text. They occur when a malevolent contributor uses the platform to copy and paste random texts from the web.

2. **Chat**. This type of errors verifies when the translation memory contributors exchange messages instead of providing translations. For example the English text "How are you?" translates in Italian as "Come stai?". Instead of providing the translation the contributor answers "Bene" ("Fine").

3. **Language Error**. This kind of errors occurs when the languages of the source or target segments are mistaken. The contributors accidentally interchange the languages of source and target segments. We would like to recover from this error and pass to the classifier the correct source and target segments. There are also cases when a different language code is assigned to the source or target segment. This happens when the parallel corpora contain segments in multiple languages (e.g. the English part of the corpus contains segments in French). The aligner does not check the language code of the aligned segments.

4. **Partial Translations**. This error verifies when the contributors translate only a part of the source segment. For example, the English source segment "Early 1980s. Muirfield C.C." is translated in Italian partially: "Primi anni 1980" ("Early 1980s").

The errors **Random Text** and **Chat** take place in the collaborative strategy of enriching MyMemory. The **Language Error** and **Partial Translations** are pervasive errors.

It is relatively easy to find positive examples because the high majority of bi-segments are correct. Finding good negative examples is not so easy as it requires reading a lot of translation segments. Inspecting small samples of bi-segments corresponding to the three methods, we noticed that the highest percentage of errors come from the collaborative web interface. To verify that this is indeed the case we make use of an insight first time articulated by Church and Gale (Gale and Church, 1993). The idea is that in a parallel corpus the corresponding segments have roughly the same length[3]. To quantify the difference between the length of the source and destination segments we use a modified Church-Gale length difference (Tiedemann, 2011) presented in equation 1 :

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

In figures 2 and 3 we plot the distribution of the relative frequency of Church Gale scores for two sets of bi-segments with source segments in English and target segments in Italian. The first set, from now on called the Matecat Set, is a set of segments extracted from the output of Matecat[4]. The bi-segments of this set are produced by professional translators and have few errors. The other bi-segment set, from now on called the Collaborative Set, is a set of collaborative bi-segments.

If it is true that the sets come from different distributions then the plots should be different. This is indeed the case. The plot for the Matecat Set is a little bit skewed to the right but close to a normal plot. In figure 2 we plot the Church Gale score obtained for the bi-segments of the Matecat set adding a normal curve over the histogram to better visualize the difference from the gaussian curve. For the Matecat set the Church Gale score varies in the interval -4.18 ...4.26.

The plot for the Collaborative Set has the distribution of scores concentrated in the center as can be seen in 3 . In figure 4 we add a normal curve to the the previous histogram. The relative frequency of the scores away from the center is much lower than the scores in the center. Therefore to get a better wiew of the distribution the y axis is reduced to the interval 0...0.1. For the Collaborative set the

[3]This simple idea is implemented in many sentence aligners.

[4]Matecat is a free web based CAT tool that can be used at the following address: https://www.matecat.com

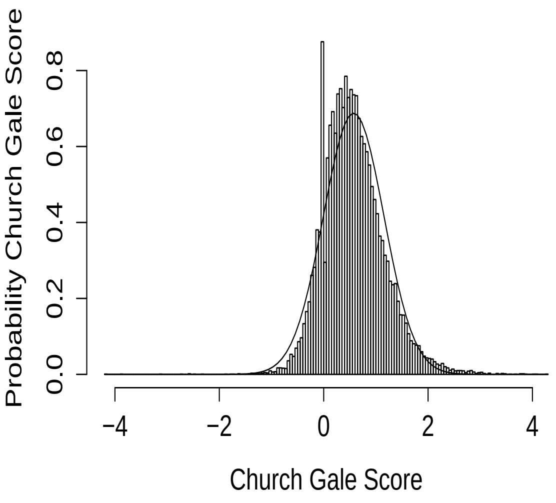

Figure 2: The distribution of Church Gale Scores in the Matecat Set

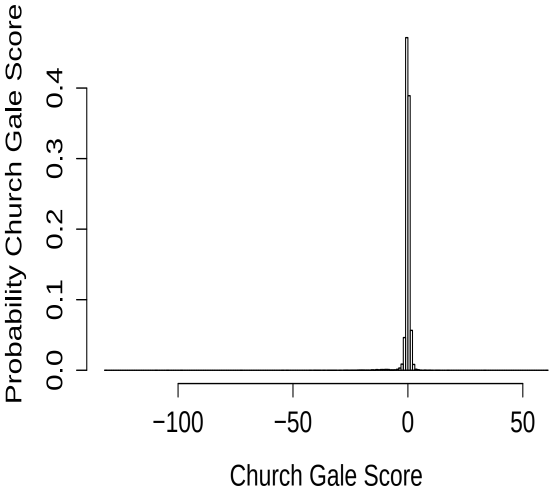

Figure 3: The distribution of Church Gale Scores in the Collaborative Set

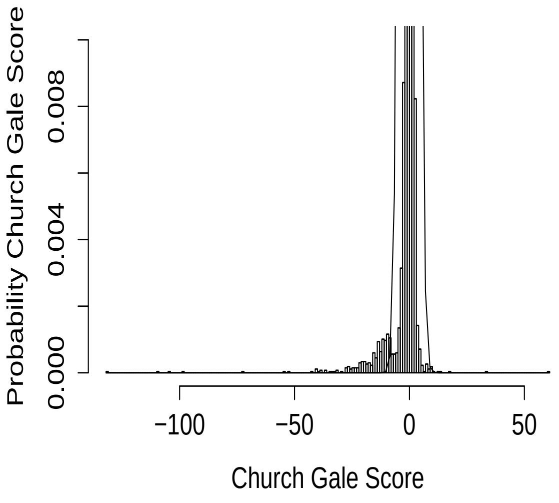

Figure 4: The normal curve added to the distribution of Church Gale Scores in the Collaborative Set

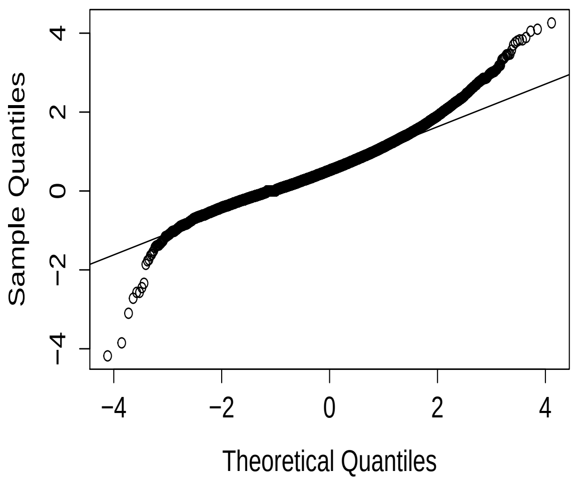

Figure 5: The Q-Q plot for the Matecat set

Church Gale score varies in the interval -131.51 ...60.15.

To see how close the distribution of Church-Gale scores is to a normal distribution we have plotted these distributions against the normal distribution using the Quantile to Quantile plot in figures 5 and 6.

In the Collaborative Set the scores that have a low probability could be a source of errors. To build the training set we first draw random bi-segments from the Matecat Set. As said before the bi-segments in the Matecat Set should contain mainly positive examples. Second, we draw random bi-segments from the Collaborative Set biasing the sampling to the bi-segments that have scores away from the center of the distribution. In this way we hope that we draw enough negative segments. After manually validating the examples we created a training set and a test set distributed as follows :

- **Training Set**. It contains 1243 bi-segments and has 373 negative example.
- **Test Set**. It contains 309 bi-segments and has 87 negatives examples.

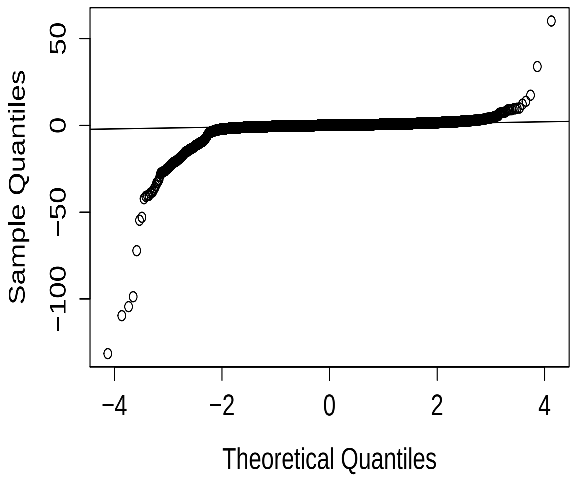

Figure 6: The Q-Q plot for the Collaborative set

The proportion of the negative examples in both sets is approximately 30%.

4 Machine Learning

In this section we discuss the features computed for the training and the test sets. Moreover, we briefly present the algorithms used for classification and the rationale for using them.

4.1 Features

The features computed for the training and test set are the following :

- *same*. This feature takes two values: 0 and 1. It has value 1 if the source and target segments are equal. There are cases specifically in the collaborative part of MyMemory when the source segment is copied in the target segment. Of course there are perfectly legitimate cases when the source and target segments are the same (e.g. when the source segment is a name entity that has the same form in the target language), but many times the value 1 indicates a spam attempt.
- *cg_score*. This feature is the Church-Gale score described in the equation 1. This score reflects the idea that the length of the source and destination segments 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 Church-Gale score is tricky because there are cases when a high Church Gale score is perfectly legitimate. For example, when the acronyms in the source language are expanded in the target language.
- *has_url*. The value of the feature is 1 if the source or target segments contain an URL address, otherwise is 0.
- *has_tag*. The value of the feature is 1 if the source or target segments contain a tag, otherwise is 0.
- *has_email*. The value of the feature is 1 if the source or target segments contain an email address, otherwise is 0.
- *has_number*. The value of the feature is 1 if the source or target segments contain a number, otherwise is 0.
- *has_capital_letters*. The value of the feature is 1 if the source or target segments contain words that have at least a capital letter, otherwise is 0.
- *has_words_capital_letters*. The value of the feature is 1 if the source or target segments contain words that consist completely of capital letters, otherwise is 0. Unlike the previous feature, this one activates only when there exists whole words in capital letters.
- *punctuation_similarity*. The value of this feature is the cosine similarity between the source and destination segments punctuation vectors. The intuition behind this feature is that source and target segments should have similar punctuation vectors if the source segment and the target segment are true translations.
- *tag_similarity*. The value of this feature is the cosine similarity between the source segment and destination segment tag vectors. The reason for introducing this feature is that the source and target segments should contain very similar tag vectors if they are true translations. This feature combines with *has_tag* to exhaust all possibilities (e.g., the tag exists/ does not exist and if it exists is present/is not present in the source and the target segments)
- *email_similarity*. The value of the feature is the cosine similarity between the source segment and destination segment email vectors. The reasoning for introducing this feature is the same as for the feature *tag_similarity*. This feature combines with the feature *has_email* to exhaust all possibilities.
- *url_similarity*. The value of the feature is the cosine similarity between the source segment and destination segment url addresses vectors. The reasoning for introducing this feature is the same as for the feature *tag_similarity*.
- *number_similarity*. The value of the feature is the cosine similarity between the source segment and destination segment number vectors. The reasoning for introducing this feature is the same as for the feature *tag_similarity*.

- *bisegment_similarity*. The value of the feature is the cosine similarity between the destination segment and the source segment translation in the destination language. It formalizes the idea that if the target segment is a true translation of the source segment then a machine translation of the source segment should be similar to the target segment.

- *capital_letters_word_difference*. The value of the 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 bi-segment. It is complementary to the feature *has_capital_letters*.

- *only_capletters_dif*. 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 the only capital letter words in the bi-segment. It is complementary to the feature *has_words_capital_letters*.

- *lang_dif*. The value of the feature is calculated from the language codes declared in the segment and the language codes detected by a language detector. For example, if we expect the source segment language code to be ”en” and the target segment language code to be ”it” and the language detector detects ”en” and ”it”, then the value of the feature is 0 (en-en,it-it). If instead the language detector detects ”en” and ”fr” then the value of the feature is 1 (en-en,it-fr) and if it detects ”de” and ”fr” (en-de,it-fr) then the value is 2.

All feature values are normalized between 0 and 1. The most important features are *bisegment_similarity* and *lang_dif*. The other features are either sparse (e.g. relatively few bi-segments contain URLs, emails or tags) or they do not describe the translation process very accurately. For example, we assumed that the punctuation in the source and target segments should be similar, which is true for many bi-segments. However, there are also many bi-segments where the translation of the source segment in the target language lacks punctuation.

The translation of the source English segment to Italian is performed with the Bing API. The computation of the language codes for the bi-segment is done with the highly accurate language detector Cybozu[5].

4.2 Algorithms

As we showed in section 3 there are cases when the contributors mistake the language codes of the source and target segments. Nevertheless, the segments might be true translations. Therefore, before applying the machine learning algorithms, we first invert the source and target segments if the above situation verifies. We tested the following classification algorithms from the package scikit-learn (Pedregosa et al., 2011):

- **Decision Tree**. The decision trees are one of the oldest classification algorithms. Even if they are known to overfit the training data they have the advantage that the rules inferred are readable by humans. This means that we can tamper with the automatically inferred rules and at least theoretically create a better decision tree.

- **Random Forest**. Random forests are ensemble classifiers that consist of multiple decision trees. The final prediction is the mode of individual tree predictions. The Random Forest has a lower probability to overfit the data than the Decision Trees.

- **Logistic Regression**. The Logistic Regression works particularly well when the features are linearly separable. In addition, the classifier is robust to noise, avoids overfitting and its output can be interpreted as probability scores.

- **Support Vector Machines** with the linear kernel. Support Vector Machines are one of the most used classification algorithms.

- **Gaussian Naive Bayes**. If the conditional independence that the naive Bayes class of algorithm postulates holds, the training converges faster than logistic regression and the algorithm needs less training instances.

- **K-Nearst Neighbors**. This algorithm classifies a new instance based on the distance it has to k training instances. The prediction output is the label that classifies the majority. Because it is a non-parametric method, it can

[5]https://github.com/shuyo/language-detection/blob/wiki/ProjectHome.md

give good results in classification problems where the decision boundary is irregular.

5 Results and discussion

We performed two evaluations of the machine learning algorithms presented in the previous section. The first evaluation is a three-fold stratified classification on the training set. The algorithms are evaluated against two baselines. The first baseline it is called Baseline Uniform and it generates predictions randomly. The second baseline is called Baseline Stratified and generates predictions by respecting the training set class distribution. The results of the first evaluation are given in table 1 :

Algorithm	Precision	Recall	F1
Random Forest	0.95	0.97	0.96
Decision Tree	0.98	0.97	0.97
SVM	0.94	0.98	0.96
K-Nearst Neighbors	0.94	0.98	0.96
Logistic Regression	0.92	0.98	0.95
Gaussian Naive Bayes	0.86	0.96	0.91
Baseline Uniform	0.69	0.53	0.60
Baseline Stratified	0.70	0.73	0.71

Table 1: The results of the three-fold stratified classification.

Excepts for the **Gaussian Naive Bayes** all other algorithms have excellent results. All algorithms beat the baselines by a significant margin (at least 20 points).

The second evaluation is performed against the test set. The baselines are the same as in three-fold evaluation above and the results are in table 2.

The results for the second evaluation are worse than the results for the first evaluation. For example, the difference between the F1-scores of the best performing algorithm: SVM and the stratified baseline is of 10%: twice lower than the difference between the best performing classification algorithm and the same baseline for the first evaluation. This fact might be explained partially by the great variety of the bi-segments in the Matecat and Web Sets. Obviously this variety is not fully captured by the training set.

Algorithm	Precision	Recall	F1
Random Forest	0.85	0.63	0.72
Decision Tree	0.82	0.69	0.75
SVM	0.82	0.81	0.81
K-Nearst Neighbors	0.83	0.66	0.74
Logistic Regression	0.80	0.80	0.80
Gaussian Naive Bayes	0.76	0.61	0.68
Baseline Uniform	0.71	0.72	0.71
Baseline Stratified	0.70	0.51	0.59

Table 2: The results of the classification on the test set.

Unlike in the first evaluation, in the second one we have two clear winners: Support Vector Machines (with the linear kernel) and Logistic Regression. They produce F1-scores around 0.8. The results might seem impressive, but they are insufficient for automatically cleaning MyMemory. To understand why this is the case we inspect the results of the confusion table for the SVM algorithm. From the 309 examples in the test set 175 are true positives, 42 false positives, 32 false negatives and 60 true negatives. This means that around 10% of all examples corresponding to the false negatives will be thrown away. Applying this method to the MyMemory database would result in the elimination of many good bi-segments. We should therefore search for better methods of cleaning where the precision is increased even if the recall drops. We make some suggestions in the next section.

6 Conclusions and further work

In this paper we studied the performance of various classification algorithms for identifying false bi-segments in translation memories. We have shown that the distribution of the Church-Gale scores in two sets of bi-segments that contain different proportion of positive and negative examples is dissimilar. This distribution is closer to the normal distribution for the MateCat set and more sparse for Collective Set. The best performing classification algorithms are Support Vector Machines (with the linear kernel) and Logistic Regression. Both algorithms produce a significant number of false negative examples. In this case the performance of finding the true negative examples does not offset the cost of deleting the false negatives from the database.

There are two potential solutions to this problem. The first solution is to improve the performance of the classifiers. In the future we will study ensemble classifiers that can potentially boost the performance of the classification task. The idea behind the ensemble classifiers is that with differently behaving classifiers one classifier can compensate for the errors of other classifiers. If this solution does not give the expected results we will focus on a subset of bi-segments for which the classification precision is more than 90%. For example, the Logistic Regression classification output can be interpreted as probability. Our hope is that the probabilities scores can be ranked and that higher scores correlate with the confidence that a bi-segment is positive or negative.

Another improvement will be the substitution of the machine translation module with a simpler translation system based on bilingual dictionaries. The machine translation module works well with an average numbers of bi-segments. For example, the machine translation system we employ can handle 40000 bi-segments per day. However, this system is not scalable, it costs too much and it cannot handle the entire MyMemory database. Unlike a machine translation system, a dictionary is relatively easy to build using an aligner. Moreover, a system based on an indexed bilingual dictionary should be much faster than a machine translation system.

Acknowledgments

The research reported in this paper is supported by the People Programme (Marie Curie Actions) of the European Unions Framework Programme (FP7/2007-2013) under REA grant agreement no. 317471.

References

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

B. L. Humphreys and D. A. Lindberg. 1993. The UMLS project: making the conceptual connection between users and the information they need. *Bull Med Libr Assoc*, 81(2):170–177, April.

Philipp Koehn and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31.

Zhifei Li and Sanjeev Khudanpur. 2009. Large-scale discriminative n-gram language models for statistical machine translation. In *Proceedings of AMTA*.

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.

Uwe Reinke. 2013. State of the art in translation memory technology. *Translation: Computation, Corpora, Cognition*, 3(1).

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2013. Dgt-tm: A freely available translation memory in 22 languages. *CoRR*, abs/1309.5226.

Jörg Tiedemann. 2011. *Bitext Alignment*. Number 14 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA, USA.

Marco Trombetti. 2009. Creating the world's largest translation memory.

Ventsislav Zhechev and Josef van Genabith. 2010. Maximising tm performance through sub-tree alignment and smt. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.