Hybrid Parallel Sentence Mining from Comparable Corpora

Dan Ştefănescu
RACAI
Calea 13 Septembrie, 13
Bucharest, Romania
danstef@racai.ro

Radu Ion
RACAI
Calea 13 Septembrie, 13
Bucharest, Romania
radu@racai.ro

Sabine Hunsicker
DFKI
Stuhlsatzenhausweg 3, 66123
Saarbrücken, Germany
sabine.hunsicker@dfki.de

Abstract

This paper presents a fast and accurate parallel sentence mining algorithm for comparable corpora called LEXACC based on the Cross-Language Information Retrieval framework combined with a trainable translation similarity measure that detects pairs of parallel and quasi-parallel sentences. LEXACC obtains state-of-the-art results in comparison with established approaches.

1 Introduction

Mining for parallel sentences in comparable corpora is much more difficult than aligning sentences in parallel corpora. Sentence alignment in parallel corpora usually exploits simple empirical evidence (turned into assumptions) such as (i) the length of a sentence is proportional with the length of its translation and (ii) the discourse flow is necessarily the same in both parts of the bi-text (Gale and Church, 1993). Thus, the extraction tools search for parallel sentences around the same (relative) text positions, making sentence alignment a much easier task when compared to kind of work undertaken here.

For comparable corpora, the second assumption does not hold. Parallel sentences, should they exist at all, are scattered all around the source and target documents, and so, any two sentences[1] have to be processed in order to determine if they are parallel or not. Also, we aim at finding pairs of quasi-parallel sentences that are not entirely parallel but contain spans of contiguous text that is parallel. Thus, finding parallel sentences in comparable corpora is confronted with the vast search space one has to consider since any positional clues indicating parallel or partially parallel sentences are not available.

The brute force approach is to analyze every element of the Cartesian product between the two sets containing sentences in the source and target languages. This approach is clearly impractical since the resulting algorithm would be very slow and/or memory consuming.[2] To reduce the search space, we turned to a framework that belongs to Information Retrieval: Cross-Language Information Retrieval (CLIR). The idea is simple: use a search engine to find sentences in the target corpus that are the most probable translations of a given sentence from the source corpus. The first step is to consider the target sentences as documents and index them. Then, for each sentence in the source corpus, one selects the content words and translates them into the target language according to a given dictionary. The translations are used to form a Boolean query which is then fed to the search engine. The top hits are considered to be translation candidates.

Using the CLIR approach to select a set of candidate target sentences (out of all target sentences) for the input source sentence is one way to dramatically reduce the search space. The reduced search space will serve another practical concern: the execution time. Thus, each candidate target sentence can be compared with the input sentence using a computationally much more complex translation similarity measure that would otherwise require an unacceptable amount of time to finish analyzing all possible pairs.

In what follows, we present our own adaptation of the hybrid CLIR/translation similarity measure approach to parallel sentence mining from comparable corpora called "Lucene-based Parallel Sentence Extraction from Comparable Corpora" (LEXACC). We describe the indexing

[1] Or a carefully selected set of sentence pairs as we will see in the next sections.

[2] With the possible exception of the parallelizing the computations but this issue is beyond the scope of this paper.

of the target corpus in subsection 3.1, the Boolean query generation for the input sentence in subsection 3.2, an additional filtering step on the output of the Lucene search engine in subsection 3.3 and our design of the translation similarity measure in section 4. We present a host of experiments aimed at assessing the performance of LEXACC from both the CLIR perspective (precision, recall and F1-measure) and practical SMT experimenting with data produced by LEXACC.

2 Related Work

Parallel data mining from comparable corpora receives its share of attention from the statistical machine translation scientific community, one of the major reasons being the fact that the Web can be seen as a vast source of comparable corpora.

The CLIR approach to finding translation candidates for sentences (reducing the search space) has received significant attention. While Rauf and Schwenk (2011) index the target sentences directly, Munteanu and Marcu (2005) index target documents for retrieving similar ones.

Another approach to cutting the search space is to perform document alignment inside the comparable corpus first and then to attempt extracting parallel sentences by inspecting the constructed document pairs only. This road has been taken by Fung and Cheung (2004) who perform document alignment using a simple dictionary-based, translation similarity measure. Recently, Ion (2011) proposes an EM algorithm that finds document alignments in a comparable corpus.

The way a pair of sentences is deemed parallel or not is usually specified with three different approaches: binary classifiers (Munteanu and Marcu, 2005; Tillman, 2009), translation similarity measures (Fung and Cheung, 2004) and generative models (Quirk et al., 2007). Our approach is somewhat similar to that of Munteanu and Marcu (2005) who used a dictionary to translate some of the words of the source sentence, and then used these translations to query a database for finding matching translation candidates. The difference resides in the fact that they choose candidate sentences based on word overlap and the decision whether a sentence pair is parallel or not is performed by a Maximum Entropy classifier trained on parallel sentences. With respect to Rauf and Schwenk (2011) who also index target sentences, our approach benefits of some filtering steps, the query is formulated using additional fields and we use a much more elaborated translation similarity measure.

3 Indexing, Searching and Filtering

3.1 Indexing target sentences

Our goal is to implement a simple yet effective solution, easily replicable. First, we split the target corpus into sentences and transform them so that we keep only stemmed non-functional words.[3] We also compute the average length in words (μ) and the standard deviation (σ) for target sentences. We consider a sentence s to be short if $length(s) \leq \mu + \sigma$ and long if $length(s) \geq \mu - \sigma$. We consider the medium-sized sentences for which $\mu - \sigma \leq \ length(s) \leq \mu + \sigma$, to be both short and long.

Following the general description presented in the introduction, we use the C# implementation of Lucene[4] to index the target sentences as Lucene documents. For each such document, we introduce three additional searchable fields, two of them corresponding to the sentence length:

(i) a field specifying if the sentence is *small*;

(ii) a field specifying if the sentence is *long*;

(iii) a field specifying the document where the target sentence belongs; this field is based on the document alignment information of the comparable corpus being processed and it is optional if such alignment information is not supplied.

3.2 Finding translation candidates for source sentences

Given an input source sentence (out of the total S source sentences), the role of the search engine is to return a list of translation candidates that are to be further analyzed. The number of hits h we take into account regulates the size of the new search space: $h * S$. The larger it is, the higher the number of candidates which can potentially increase the recall but also the computational complexity. For each sentence in the source corpus, we generate a Lucene query as follows:

(i) We employ a GIZA++ (Och and Ney, 2000) dictionary previously created from existing parallel documents. This dictionary is expected to be small due to the lack of necessary resources. For each content word we keep the best 50 translation equivalents, which are also content words, having translation probabilities above 0.1. Each of them is stemmed and added as an disjunctive query term (SHOULD occur);

(ii) We add two disjunctive query terms (SHOULD occur) standing for the length of the source sentence: *short* and *long*. Each of these

[3] We keep functional words lists for all languages.

[4] http://incubator.apache.org/lucene.net/download.html

terms can be boosted according to the importance one wants to give to matching source and target lengths. In our implementation, the value of the boosting factor is 2;

(iii) We add a compulsory query term (MUST occur) specifying the target document where the source sentence translation should be searched. However, this term can be added only if the document alignment information exists and it has been used at index creation as well.

After the query is constructed, we use it to interrogate the default Lucene search engine (no modifications on the relevance method) in order to get the best *h* hits.

3.3 Filtering

The filtering step is designed to further reduce the new search space, selecting only the best candidates for the final stage in which the translation similarity measure (Section 4) is applied. Filtering must be very fast and good enough not to filter out parallel data. We do this by computing a viability score for each candidate sentence pair and then keeping only those above the average. For a candidate pair formed by a source sentence *s* and a target sentence *t*, the formula is:

$$viabilityScore = \alpha * \beta * se * sim \qquad (1)$$

where *se* represents the score returned by the search engine and *sim* is a similarity score we will come back to later. The other factors are aiming at favoring high scores for sentences with similar (α) and large (β) lengths. In our implementations they are computed as:

$$\alpha = 1 - \frac{abs(|s| - |t|)}{\max(|s|, |t|)} \qquad (2)$$

$$\beta = \frac{\min(|s|, |t|)}{\lambda} \qquad (3)$$

where *abs* is the absolute value, |*s*| is the length in words of sentence *s* and λ is an integer constant representing the length threshold from which we consider a sentence to be very long (λ=100 in our implementation, but it can be chosen depending on the given corpora).

The similarity score (*sim*) from equation 1 is calculated according to the formula:

$$sim = \frac{2 * teFound * te}{|s| + |t|} * \frac{1}{\sqrt{coh}} \qquad (4)$$

where *teFound* is the total number of words in *s* for which we found translation equivalents in *t*, *coh* is the *cohesion score* computed as the average distance between the sorted positions of the translation equivalents found in *t* (the lower the better)[5] and *te* is calculated as:

$$te(s,t) = \sum_{w_s \in s} \max_{w_t \in t} dicScore(w_s, w_t) \qquad (5)$$

where *dicScore* is the translation probability score from the dictionary. The rationale behind equation 5 is induced by the assumption that a word w_s is translated by only one word w_t and so, $dicScore(w_s,w_t) \geq dicScore(w_s,w_i)$ for any w_i in *t*.

We should note that since we aim at gathering parallel data which is not already in the dictionary with started with, we are more interested in finding long parallel texts. It is more probable that such texts would contain (beside already known translations) unknown parallel data.

4 The Translation Similarity Measure

The binary classifier of Munteanu and Marcu (2005) associate a confidence probability with its decision but setting this confidence at 0.5 or 0.7 as they do, is equivalent to saying that sentence pairs with a score below the confidence level are not interesting for SMT.[6] Our view is that whatever sentence pairs actually improve the output of an SMT system are important and we found that these range from parallel, quasi-parallel to strongly comparable.

We modeled our translation similarity measure as a weighted sum of feature functions that indicate if the source piece of text is translated by the target. Given two sentences s in the source language and t in the target language, then the translation similarity measure $P(s,t)$ is

$$P(s,t) = \sum_i \theta_i f_i(s,t) \qquad (6)$$

such that $\sum_i \theta_i = 1$. Each feature function $f_i(s,t)$ will return a real value between 0 (s and t are not related at all) and 1 (t is a translation of s) and contributes to the overall parallelism score with a specific fraction θ_i that is language-pair dependent and that will be automatically determined by training a logistic regression classifier on existing parallel data (see next subsection).

Each of the feature functions $f_i(s,t)$ has been designed to return a value close to 1 on parallel s and t by manually inspecting a fair amount of parallel examples in the English-Romanian pair of languages. By negation, we assume that the

[5] We experimented with different power values for the cohesion score. For ½ (the square root) we had the best results.
[6] But we acknowledge the fact that the probability of a sentence pair being parallel as computed by the classifier of Munteanu and Marcu is a proper model of parallelism

same feature functions will return a value close to 0 for non-parallel, not-related s and t but this behavior is critically influenced by the quality and completeness of the linguistic computational resources that we use: bilingual translation lexicons, lists of inflectional suffixes used for stemming and lists of stop-words. Thus, generally, a feature function that uses one (or more) of the resources mentioned above can falsely return a value close to 0 for parallel s and t due to the fact that this decision was made in the absence of the relevant entries in that resource. The prototypical example here is that the translation lexicon does not contain the relevant translations for the words in s.

4.1 Features

Before being processed, sentences s and t are tokenized, functional words are identified and content word are stemmed using language-dependent inflectional suffixes. Given these transformations of s and t, all features $f_i(s,t)$ are language-independent. We use 5 features.

$f_1(s,t)$ is the "**content words translation strength**" feature. Given a statistical translation dictionary obtained by e.g. applying GIZA++ on a parallel corpus,[7] we find the best 1:1 alignment A between content words in s and t such that the translation probability [8] is maximized. If $\langle cw_i^s, cw_j^t\rangle$ is a word pair from A, $p(\langle cw_i^s, cw_j^t\rangle)$ is the translation probability of the word pair from the dictionary and $|s|$ is the length (in content words) of sentence s, then

$$f_1(s,t) = \frac{\sum_{\langle cw_i^s, cw_j^t\rangle \in A} p(\langle cw_i^s, cw_j^t\rangle)}{|s|} \quad (7)$$

This feature has a maximum value of 1 if all content words from s are translated in t with the maximum probability of 1.

$f_2(s,t)$ is the "**functional words translation strength**" feature. The intuition is that functional words around content words aligned as in ture$f_1(s,t)$, will also align for parallel s and t because of the fact that, from a dependency-syntactic point of view, functional words (prepositions, determiners, articles, particles, etc.) are usually governed by or govern nearby content words. Mathematically, if $\langle fw_k^s, fw_l^t\rangle$ is the highest scored pair of aligned functional words near (in a window of ±3 words) the aligned pair of content words $\langle cw_i^s, cw_j^t\rangle$ from A, $|A|$ is the cardinal of the best alignment as found by $f_1(s,t)$ and $p(\langle fw_k^s, fw_l^t\rangle)$ is the probability of the functional word pair from the dictionary, then

$$f_2(s,t) = \frac{\sum_{\langle cw_i^s, cw_j^t\rangle \in A} p(\langle fw_k^s, fw_l^t\rangle)}{|A|} \quad (8)$$

The maximal value of $f_2(s,t)$ is 1 and it is reached when for each aligned pair of content words from A, there is a pair of functional words that align with the maximum probability of 1.

$f_3(s,t)$ is the "**alignment obliqueness**" feature (Tufiş et al., 2006). Here we have redefined it to be a discounted correlation measure because there are pairs of languages for which the natural word order implies crossing word alignment links. $f_3(s,t)$ also uses the alignment set A of content words described for feature $f_1(s,t)$ from which we derive two source and target vectors x^s and x^t of the same length containing the indices i in the ascending order ($1 \le i \le |s|$) and j respectively ($1 \le j \le |t|$) of content words cw_i^s and cw_j^t that form an alignment pair in A. Alignment obliqueness is computed as

$$f_3(s,t) = \mathrm{abs}(\rho_{x^s,x^t}) \frac{1}{1+e^{-10\frac{|A|}{\min(|s|,|t|)}+5}} \quad (9)$$

where ρ_{x^s,x^t} is the Pearson correlation coefficient of the x^s and x^t vectors and $\mathrm{abs}(x)$ is the absolute value function. The second term is a modified sigmoid function $f(x) = \frac{1}{1+e^{-10x+5}}$ designed to be a discount factor with values between 0 and 1 when x takes on values between 0 and 1. The rather steep variation of $f(x)$ was experimentally modeled in order to heavily discount "rare" alignments for which the Pearson correlation is high. Thus, if A contains only a few alignments relative to $\min(|s|,|t|)$ (the size of A is at most $\min(|s|,|t|)$), then even if ρ_{x^s,x^t} is high, $f_3(s,t)$ should be small because a few alignments usually do not indicate parallelism.

$f_4(s,t)$ is the "**strong translation sentinels**" feature. Intuitively, if sentences s and t are parallel then, frequently (at least in our studied examples), one can find content words that align near the beginning and end of the considered sentences. $f_4(s,t)$ is a binary-valued feature which is 1 if we can find "strong" translation pairs (probability greater than 0.2; set experimentally) between the first 2 content words at the beginning

[7] To obtain the dictionaries mentioned throughout this section, we have applied GIZA++ on the JRC Acquis corpus (http://langtech.jrc.it/JRC-Acquis.html).

[8] For two source and target words, if the pair is not in the dictionary, we use a 0 to 1 normalized version of the Levenshtein distance in order to assign a "translation probability" based on string similarity alone. If the source and target words are similar above a certain threshold (experimentally set to 0.7), we consider them to be translations.

of s and t and between the last 2 content words at the end of s and t. $f_4(s,t)$ is 0 otherwise.

Finally, $f_5(s,t)$ is the "**end with the same punctuation**" feature. This is also a binary-valued feature which is 1 if both s and t end with the same type of punctuation: period, exclamation mark, etc. It is also 1 if both s and t lack final punctuation. $f_5(s,t)$ is 0 otherwise.

The observant reader has noticed by now that all the features with the exception of $f_5(s,t)$ are not symmetrical because they all depend on the alignment A computed for $f_1(s,t)$ which is not symmetrical and as such, the measure from equation 6 is not symmetrical as well. In order to have evidence from both directions, we will use the arithmetic mean to get the final measure:

$$M(s,t) = M(t,s) = \frac{P(s,t) + P(t,s)}{2} \qquad (10)$$

4.2 Learning the optimal weights

The weights $\theta_2$and θ_3 corresponding to the features "functional words translation strength" and "alignment obliqueness" are language-pair dependent because of the specific word ordering of the source and target languages. At the same time, θ_1 through θ_4 have to be optimized with respect to the translation lexicon in use, since the construction of the word alignments is based on this dictionary. Also, since $P(s,t)$ is not symmetrical, we will have to learn different θ_i weights from source to target and vice versa.

In order to derive a set of optimal weights for each language pair and translation lexicon, we have trained a standard logistic regression classifier. Briefly, the logistic regression classifier learns the θ_i weights that define the hyperplane (whose equation is the same as equation 6) that best separates the positive training examples from the negative ones. In our case, the examples are the multidimensional points whose coordinates are given by the feature functions $f_i(s,t)$.

For each language pair, the training set consists of 9500 parallel sentences[9] for the positive examples and 9500 of non-parallel sentences (obtained from the parallel pairs by random shuffling) for the negative examples. For the training set in question, we also have 500 additional parallel sentences together with 500 non-parallel sentences (obtained by random shuffling as well) as the test set. An example[10] is obtained by computing all the feature functions $f_i(s,t)$ for the given positive (parallel) or negative (non-parallel) s and t.

Table 1 summarizes the derived optimal weights for 8 language-pairs, in both directions. In every pair, one language is English (en) and the others are: Croatian (hr), Estonian (et), German (de), Greek (el), Lithuanian (lt), Latvian (lv), Romanian (ro) and Slovene (sl).

Lang.	θ_1	θ_2	θ_3	θ_4	θ_5	**F1/BL**
en–ro	0.31	0.02	0.37	0.21	0.09	0.93/0.88
ro–en	0.31	0.01	0.37	0.20	0.11	0.93/0.91
en–de	0.31	0.02	0.3	0.17	0.2	0.94/0.89
de–en	0.35	0.02	0.28	0.16	0.19	0.96/0.92
en–sl	0.23	0.01	0.38	0.2	0.18	0.96/0.89
sl–en	0.2	0.03	0.38	0.19	0.2	0.94/0.89
en–el	0.61	0.08	0.21	0	0.1	0.99/0.98
el–en	0.47	0.08	0.28	0.07	0.1	0.98/0.98
en–lv	0.27	0.05	0.41	0.16	0.1	0.98/0.96
lv–en	0.49	0.03	0.41	0	0.07	0.99/0.96
en–lt	0.33	0.01	0.41	0.15	0.1	0.96/0.91
lt–en	0.28	0.01	0.41	0.15	0.15	0.94/0.90
en–et	0.28	0.08	0.36	0.17	0.11	0.98/0.96
et–en	0.27	0.07	0.38	0.18	0.1	0.96/0.93
en–hr	0.29	0.01	0.41	0.16	0.13	0.98/0.95
hr–en	0.25	0.02	0.44	0.17	0.12	0.98/0.97

Table 1: Optimal weights for the translation similarity measure

The column named "F1/BL" (see Table 1) indicates the gain in F1 measure when testing the translation similarity measure with the optimal weights on the test set as compared to a baseline (BL) consisting of applying the measure using fixed values of the weights corresponding to our intuition of their importance: $\theta_1 = 0.45$, $\theta_2 = 0.2$, $\theta_3 = 0.15$, $\theta_4 = 0.15$, $\theta_5 = 0.05$. For instance, we imagined that the content words translation strength feature $f_1(s,t)$ is much more important compared to the rest of the features but the training procedure proved us wrong.

5 Experiments and Results

5.1 Experiment Setting

We evaluated our approach on 7 pairs of languages under the framework of the ACCURAT project.[11] For each pair, the source language is English (en), while the target languages are: Estonian (et), German (de), Greek (el), Lithuanian (lt), Latvian (lv), Romanian (ro) and Slovene (sl). In order to compute precision and recall when mining for parallel sentences, we have devised

[9] Mostly from the News domain for all language pairs.

[10] When an example occurs multiple times with both labels, we retain all the occurrences of the example with the most frequent label and remove all the conflicting occurrences.

[11] http://www.accurat-project.eu/

artificial comparable corpora for all mentioned language pairs, with different levels of controlled comparability. Starting from 100 news parallel sentences for all language pairs, the corpora were created by injecting noise (in specific proportions) extracted from the News corpora collected in the ACCURAT project. We experimented with 4 different amounts of noise: 2:1,[12] 5:1, 10:1, 100:1, corresponding to different degrees of comparability, from strongly comparable to weakly comparable. The worst case scenario is by far the one with 100:1 noise and so, most of our experiments were developed under this setting.

We evaluated the efficiency of LEXACC after each of its steps: (i) the extraction of translation pair candidates using the search engine, (ii) candidate pairs filtering and (iii) the usage of the translation similarity measure. Moreover, we evaluated the impact of the extracted data when used for improving SMT translation models.

5.2 Search Engine Efficiency

To measure the efficiency of using the search engine for finding translation candidates in the worst case scenario (100:1 noise ratio), we computed the recall we would obtain if we would have kept the best 100 hits (target sentences) returned by the engine for each source sentence. Instead of brute force analyzing $10{,}100^2$ sentence pairs, we can now look at only 1 million pairs. This means a search space reduction of about 100 times. Table 2 shows that this approach is effective for most of the language pairs, but poor for en–el and en–ro. One of the reasons might be the quality of the dictionaries we relied on when generating the search engine queries.

Pair	Recall UB	Data Size (pairs / disk size)
en–de	0.98	1,009,500 / 323 Mb
en–el	0.42	1,009,700 / 485 Mb
en–et	0.89	1,008,800 / 345 Mb
en–lt	0.93	1,008,200 / 350 Mb
en–lv	0.92	1,008,300 / 366 Mb
en–ro	0.69	1,009,800 / 294 Mb
en–sl	0.80	688,266 / 191 Mb

Table 2: Recall upper boundary (UB) and size (sentence pairs and disk space occupied) for the translation candidates returned by Lucene

5.3 Filtering Efficiency

As already mentioned, filtering is an intermediary step designed to further reduce the search space used for the final analysis. The filtering module receives high scores for speed and search space reduction for all language pairs. However, in terms of preserving the recall upper boundary, it performs well only for en–lv and en–de and acceptable for en–ro and en–el. It loses about 40% recall for the other 3 language pairs. Table 3 summarizes the results.

Pair	Recall UB	Recall Loss	Size (pairs / disk size)	Search Space Drop
en–de	0.83	15.30%	20,868 / 10 Mb	97.93%
en–el	0.30	28.57%	108,629/69 Mb	89.24%
en–et	0.54	39.32%	34,051 / 22 Mb	96.62%
en–lt	0.57	38.70%	35,831 / 21 Mb	96.44%
en–lv	0.83	9.78%	91,305 / 45 Mb	90.94%
en–ro	0.53	23.18%	160,968/67 Mb	84.05%
en–sl	0.44	45%	65,191 / 28 Mb	90.52%

Table 3: Recall upper boundary and size after the filtering step

5.4 Translation Similarity Efficiency

We evaluated the efficiency of the Translation Similarity Measure (TSM) from Section 4 by comparing it with the MaxEnt classifier by Munteanu and Marcu (2005) on English-German (en–de) document pairs with different levels of comparability (2:1 noise ratio, 5:1 and 10:1; see section 5.1). For both TSM and MaxEnt (with the associated confidence score for the "parallel" label), we took into account all possible thresholds with a granularity of 0.01 above which the candidate pairs are considered parallel. We report the results corresponding to the threshold that maximizes F1 for TSM and F1 for MaxEnt (threshold are not the same). We explored 3 possible scenarios. The first one (Table 4) is to compute TSM for all possible sentence pairs.

	2:1		5:1		10:1	
	ME	**TSM**	**ME**	**TSM**	**ME**	**TSM**
P	0.800	0.791	0.789	0.760	0.523	0.724
R	0.560	0.760	0.450	0.700	0.450	0.630
F1	0.658	0.775	0.573	0.729	0.483	0.673

Table 4: en–de comparison between the MaxEnt classifier (ME) and the TSM when applied individually onto all possible sentence pairs

The second scenario (Table 5) is to compute TSM only for the candidate pairs proposed by the search engine, without filtering.

	2:1		5:1		10:1	
	ME	**LEX**	**ME**	**LEX**	**ME**	**LEX**
P	0.800	0.717	0.789	0.650	0.523	0.618
R	0.560	0.710	0.450	0.650	0.450	0.600
F1	0.658	0.713	0.573	0.650	0.483	0.609

Table 5: en–de comparison between the MaxEnt classifier and LEXACC with no filtering

[12] For each parallel sentence, 2 noise sentences were added

The third scenario is similar to the second one, only this time we use filtering.

	2:1		5:1		10:1	
	ME	**LEX**	**ME**	**LEX**	**ME**	**LEX**
P	0.800	0.809	0.789	0.737	0.523	0.742
R	0.560	0.340	0.450	0.450	0.450	0.520
F1	0.658	0.478	0.573	0.559	0.483	0.611

Table 6: en–de comparison between the MaxEnt classifier and LEXACC with filtering

For strongly comparable corpora (with less noise, like the 2:1 corpus) the filtering step is in fact worsening the results. This is something to be expected because the filtering step eliminates a large proportion of the candidate pairs returned by the engine. Thus, filtering should be used only for weakly comparable corpora.

In order to make things more clear, we performed yet another experiment, this time for 100:1 noise ratio which corresponds to a very weakly comparable corpus. In this setting, taking into account all possible sentence pairs as candidate pairs would result in a huge running time and so, we were able to compare only the results obtained by LEXACC with and without filtering.

	LEXACC NO filtering		LEXACC WITH filtering
	Best	**Same T**[13]	**Best**
P	0.327	0.101	0.800
R	0.370	0.710	0.640
F1	0.347	0.177	0.711
Threshold	*0.59*	*0.41*	*0.41*
Running Time	*49.72 minutes*		*5.53 minutes*

Table 7: En-De comparison between LEXACC with and without filtering for 100:1 noise

We can see that for weakly comparable corpora, at the same threshold (0.41), filtering gets rid of a lot of noise, keeping the precision high (compare 0.8 with 0.101) at a modest decrease of the recall (compare 0.64 with 0.71).

Table 8 shows the accuracy of LEXACC when running on the 100:1 noise ratio comparable corpora. The running times depend on the sentence lengths and the size of the dictionaries.

Pair	P	R	F1	Thr.	Minutes
en–de	0.800	0.64	0.711	0.41	5.53
en–el	0.550	0.22	0.314	0.35	27.24
en–et	0.284	0.23	0.254	0.34	7.11
en–lt	0.398	0.41	0.403	0.39	8.24
en–lv	0.357	0.50	0.416	0.51	11.75
en–ro	0.473	0.27	0.343	0.65	37.33
en–sl	0.219	0.16	0.185	0.34	7.75

Table 8: LEXACC (with filtering) run on the 100:1 noise ratio comparable corpora

[13] Same T: results obtained without filtering for the threshold yielding the best results with filtering (0.41).

5.5 SMT Experiments

To test the quality of the data extracted by LEXACC, we ran a few experiments with domain-adapted SMT in the automotive industry domain. We manually created a parallel corpus from an English-German comparable corpus of about 3.5 million sentences per language collected from the Web. The results of the experiments with the LEXACC extracted data were compared to the same experiments conducted with the manually extracted parallel data, to examine and compare the influence of the LEXACC extracted data. Table 9 shows the statistics on the sentence pairs and sentence counts in the parallel and LEXACC extracted data.

Data	#pairs	# unique sent. (de/en)
parallel	44,482	42,396 / 44,290
extracted	45,952	12,718 / 13,306

Table 9: Statistics on parallel and extracted data

We compared three systems in our experiments: the "Baseline" system which was trained only on the Europarl (EP, (Koehn, 2005)) and News Commentary corpus (NC), [14] "Automotive.parallel" which added only the parallel data to the baseline and the "Automotive.extracted" which added only the LEXACC extracted data to the baseline. All resulting corpora were aligned using GIZA++ and the MT systems were trained using the Moses SMT Toolkit (Koehn et al., 2007). The languages models were trained using SRILM (Stolcke, 2002).

The Baseline system only uses Europarl, both for the translation and the language model but for the two adapted systems we used an additional language model trained on the domain-specific texts. Tuning via MERT was performed for all systems on a domain-specific development set; testing also used text from the automotive domain. The translations were evaluated using BLEU (Papineni et al., 2001).

System	BLEU
Baseline	18.81%
Automotive.parallel	30.25%
Automotive.extracted	25.44%

Table 10: BLEU scores

As Table 10 shows, it is possible to gain about 6.5 BLEU points over the baseline system with the extracted data. The parallel data outperforms LEXACC, which may be due to the fact that the parallel data includes more unique sentences (see Table 9). But although only approx. 30% of the available unique data was extracted, an increase

[14] http://www.statmt.org/wmt11/translation-task.html

of 6.5 BLEU points is recorded -- more than half of the increase achieved with the full parallel data. This means that LEXACC is able to discover salient parallel data that brings significant gains in BLUE score despite its size.

Another area of interest is how the extracted parallel and strongly comparable data compares to clean parallel data. In the extracted data, every German sentence is linked to 3.5 English sentences on average. To examine the effect of this noise, we retrained "Automotive.parallel" with increasing amounts of data. Table 11 shows that the extracted data corresponds to more than 15k of parallel data in terms of BLEU improvement.

System	Training Data	BLEU score
Baseline	EP+NC	18.81%
Automotive.5k	EP+NC+5k Automotive	22.02%
Automotive.10k	EP+NC+10k Automotive	23.36%
Automotive.15k	EP+NC+15k Automotive	24.98%
Automotive.20k	EP+NC+20k Automotive	26.48%
Automotive.45k	EP+NC+full Automotive	30.25%

Table 11: Experiments with adding data

The data LEXACC extracts is of high enough quality to be useful for SMT purposes, as the noise is filtered out during the training phase.

6 Conclusions

Parallel sentence mining from comparable corpora is a well-studied problem with several reliable solutions already discussed in the literature. We present yet another original hybrid approach (LEXACC) based on CLIR combined with a complex, trainable translation similarity measure but with a strong emphasis on practical issues such as the reduction of the search space and the behaviour of the translation similarity measure as a function of the comparability level of the corpus (an aspect that is not well studied).

LEXACC is currently used in the ACCURAT project for parallel data mining from comparable corpora and we have presented evidence that it is able to extract good quality parallel sentences that improve SMT systems.

7 Acknowledgements

This work has been supported by the ACCURAT project (http://www.accurat-project.eu/) funded by the European Community's Seventh Framework Program (FP7/2007-2013) under the Grant Agreement no. 248347.

References

Fung, Pascale and Percy Cheung. 2004. *Mining very-non-parallel corpora: parallel sentence and lexicon extraction via bootstrapping and EM*. In: Proceedings of the EMNLP-2004, Barcelona, Spain, pp. 57–63.

Gale, William A. and Kenneth W. Church. 1993. *A Program for Aligning Sentences in Bilingual Corpora*. Computational Linguistics 19 (1): 75–102.

Ion, Radu, Alexandru Ceauşu and Elena Irimia. 2011. *An Expectation Maximization Algorithm for Textual Unit Alignment*. In: Proceedings of BUCC-2011, Portland, Oregon, USA, pp. 128—135.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation*. In: Proceedings of the 45th Annual Meeting of the ACL Companion Volume Proceedings of the Demo and Poster Sessions, Prague, pp. 177–180.

Koehn, Philipp. 2005. *Europarl: A Parallel Corpus for Statistical Machine Translation*. In: Proceedings of MT Summit 2005.

Munteanu, Dragos and Daniel Marcu. 2005. *Improving Machine Translation Performance by Exploiting Comparable Corpora*. Computational Linguistics, 31(4): 477–504.

Och, Franz Josef and Hermann Ney. 2000. *Improved Statistical Alignment Models*. In: Proceedings of the ACL 2000, Hong Kong, China, pp. 440– 447.

Papineni, Kishore, Salim Roukos, Todd Ward and-Wei-Jing Zhu. 2001. *Bleu: a Method for Automatic Evaluation of Machine Translation*. IBM Report.

Quirk, Chris, Raghavendra Udupa U. and Arul Menezes. 2007. *Generative Models of Noisy Translations with Applications to Parallel Fragment Extraction*. In: Proceedings of the MT Summit XI, European Association for Machine Translation.

Rauf, Sadaf and Holger Schwenk. 2011. *Parallel sentence generation from comparable corpora for improved SMT*. Machine Translation, 25(4): 341–375.

Stolcke, Andreas. 2002. *SRILM - An Extensible Language Modeling Toolkit*. In: Proceedings of ICSLP, Vol. 2, pp. 901–904.

Tillmann, Christoph. 2009. *A Beam-Search Extraction Algorithm for Comparable Data*. Proceedings of the ACL-IJCNLP 2009 Conference Short Papers.

Tufiş, Dan, Radu Ion, Alexandru Ceauşu and Dan Ştefănescu. 2006. *Improved Lexical Alignment by Combining Multiple Reified Alignments*. Proceedings of EACL 2006, Trento, Italy, pp. 153–160.