Better handling of a bilingual collection of texts

Master's thesis

Alexandre Bérard

Supervisor: Philippe Langlais

RALI
Université de Montréal
Canada
June 11, 2014

Abstract

Statistical machine translation models are trained from parallel corpora, which are collections of translated texts. These texts are usually processed using dedicated tools called "sentence aligners", which output parallel sentence pairs. However, parallel resources are very scarce in certain languages or domains. Alternative solutions have been proposed that extract parallel sentences from the so-called "comparable corpora", containing texts in different languages sharing similar topics. But comparable corpora can contain document pairs with various degrees of parallelism. For example, in the Wikipedia corpus, many article pairs are actually parallel. We implement a system to extract parallel sentences from comparable corpora, and apply this system on the Wikipedia corpus. We also propose a method that determines whether two documents are parallel. By comparing sentence aligners with our parallel sentence extraction system, we suggest that extracting the parallel document pairs in a comparable corpus and using a sentence aligner on them might help improve the recall.

Contents

1 Introduction

With the coming of the era of Big Data, never has there been more digital data available to computers. A whole new dimension has been unlocked for natural language processing applications. Machine translation, in particular, has reached its peak, with the so-called *Statistical Machine Translation* (SMT). While older translation systems made use of handcrafted rules, or expensive linguistic resources, and were restricted to a handful of domains and language pairs, SMT models translation in a very simple and general way, which makes these systems easily portable to new languages and domains, while offering unequaled performance.

SMT models are automatically trained from large collections of translated texts. Such sets are called *parallel corpora*.

A parallel corpus contains texts paired with their translation in another language. Most of the parallel resources that are freely available come from the governments of multilingual countries or international institutions. Examples of such resources are the Europarl corpus, which is a collection of the proceedings of the EU parliament (available in 21 languages), or the United Nations proceedings, or the Canadian Hansards, produced by the Canadian government (French and English). But other resources include Web page translations, manuals of open source software, translated News articles, etc. Parallel corpora are usually processed by *sentence aligners*, which are tools that align translated texts at the sentence level, to find pairs of sentences, which can then be used to train SMT models.

Even though SMT has been improved over the years, it is still limited by the lack of available resources in certain language pairs and domains. Most parallel resources come from the political domain, which results in poor translation quality in other domains. Similarly, while very large amounts of parallel corpora exist for the French-English language pair, not all language pairs are as lucky.

While a certain amount of effort has been focused on producing new parallel resources, for instance by crawling the Web to find translated pages, other contributions have brought a new dimension to the problem: What if we could build new parallel corpora by finding parallel sentences in texts that are not necessarily translations? Munteanu and Marcu have proposed a system which pairs together documents on similar topics (*comparable* documents), and extracts the potential parallel sentence pairs.

The first step of our job will be to implement such a system, and to evaluate its performance under various testing conditions. In particular, we want to compare its performance with a traditional sentence aligner. The next step will be to apply this sentence extraction system on Wikipedia.

Wikipedia is a very rich comparable corpus, covering a wide range of languages. Wikipedia articles in different languages that cover similar topics are already paired, thanks to Wikipedia's "interlanguage links". [Smith et al., 2010] extract parallel sentence pairs from Wikipedia, using a method similar to [Munteanu and Marcu, 2005]. Wikipedia's article pairs exhibit various degrees of parallelism. While most pairs are not parallel at all, or contain very few sentences that are translations, many contributors enrich the encyclopedia in other languages by translating whole articles or parts of articles that already exist in a given language (e.g., English). This results in many article pairs which are actually parallel, or mostly parallel. Parallel sentence extraction methods, while they seem to be a good choice on comparable texts, do not offer the same performance as the usual sentence aligners when applied on parallel texts. Finding out which Wikipedia articles are parallel, and using a sentence aligner to process them, might indeed by a good way to improve the recall.

The next section describes how statistical machine translation works, and why it needs parallel data. The third section outlines the state of the art methods in the areas of sentence alignment and parallel sentence extraction. The fourth section describes the details of our implementation of Munteanu and Marcu's sentence extractor, that we call MUNT. Next, we perform more in-depth experiments on MUNT, in particular we measure the impact of the parallelism degree of the input corpus. In the sixth section, we extract and pair articles from Wikipedia to build a comparable corpus, and we apply MUNT on this corpus to build a *parallel corpus*. We train a SMT model with this corpus to show its usefulness to translation. Finally, we propose a system that determines whether or not two documents are parallel.

2 Statistical machine translation

2.1 Definition

The concept of statistical machine translation was first introduced in 1949 by *Waven Weaver*, who saw translation like a problem of cryptography:

> One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.'

[Brown et al., 1993] brought this idea to life, by creating the famous *IBM models*. Say we want to build a system that translates from French to English. Using *Bayes' theorem*, the conditional probability of translating a French sentence f into an English sentence e can be expressed as follows:

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)} \tag{1}$$

The goal of the translation system is to find the English sentence e which maximizes the conditional probability $P(e|f)$. The probability, $P(f)$ is independent of e, we thus arrive at the *fundamental equation of SMT*:

$$e = arg \max_{e \in e*} P(e)P(f|e) \tag{2}$$

Where $e*$ is the set of all possible English strings.

This is an instance of the *noisy channel model*, which is also used in other domains related to natural language processing (e.g. speech recognition). The problem was formulated by [Brown et al., 1993] as follows:

> We further take the view that when a native speaker of French produces a string of French words, he has actually conceived of a string of English words, which he translated mentally. Given a French string f, the job of our translation system is to find the string e that the native speaker had in mind when he produced f.

The problem is thus reversed by the noisy channel model: instead of modeling the probability $P(e|f)$ that a French sentence f produces an English sentence e, the probability $P(f|e)$ measures the likelihood that the native speaker produced the output string f, given the English string e that he had originally in mind.

The advantage of modeling machine translation in this manner, is that it splits the problem in two distinct problems: the generation of fluent English language, and the translation from English to French. The probability $P(e)$ is the *language model probability*. It ensures that the output is fluent English. $P(f|e)$ is the *translation model probability*. The job of the translation model is to find an adequate translation, regardless of its fluency. An important challenge in SMT is to build an efficient search algorithm, which finds the English sentence that maximizes the product of these two probabilities (the *decoding* algorithm).

2.2 Language model

The target language model is completely independent of the source language, which means that it can be trained using monolingual data in the target language (English in our case). Such data is much more common than bilingual data. The language model is thus much better informed about the English language than any translation model trained with bilingual data could be.

The probability of a word sequence $w_1, w_2, \ldots, w_n$ can be expressed as follows:

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2|w_1) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) \tag{3}$$

Most language models use a *n-gram* approximation, which results in a probability of:

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-k+1}, \cdots, w_{i-1}) \tag{4}$$

Where k is the order of the language model. The language model makes the assumption that the probability of observing a word at a given position, only depends on the $k-1$ previous words. It is thus easier to estimate the probability of a new sequence, than when considering the entire context. For example, in a trigram model, the conditional probability $P(w_i|w_{i-2}, w_{i-1})$ can be estimated as follows:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} \tag{5}$$

$count(w_i, \ldots, w_k)$ is the number of occurrences of the sequence $w_i, \ldots, w_k$ in the training data. Because the vocabulary size is unlimited, and the training data is not, it is very common to encounter new n-grams. We do not want the total probability of the sentence "Jacques Cartier was born in 1491 in Saint-Malo, the port on the north-west coast of Brittany." to amount to zero, because the trigram "1491 in Saint-Malo" was never seen before. That is why, in practice, more complex models are used (like *back-off models*), that attribute non-null probabilities to unseen n-grams.

Statistical machine translation generally uses language models of order *5*. The higher the order of the language model, the more accurately it models language, but the more difficult it is to generalize from a given amount of training data. A language model with large *n-grams* equals to learning by rote the sentences of the training data.

2.3 IBM models

The reason why machine translation needs parallel data, is for training the *translation model*. [Brown et al., 1993] propose 5 different word-based translation models, that they call *IBM models*. IBM models are based on a word-level alignment of the sentences.

In word-based translation, each source word may generate several target words. A word-level alignment is a one-to-many relation $j \rightarrow a_j$ between the source sentence and the target sentence, which defines by which source word, each target word was generated. To allow the insertion of target words which are not actually aligned to any source word, the source word `NULL` is added at the position 0.

Remember that our system is translating from French to English, but the noisy channel model reverses the translation direction. In the translation model, the source language is English and the target language is French. Figure 1 shows an example of word-alignment of a French sentence with an English sentence.

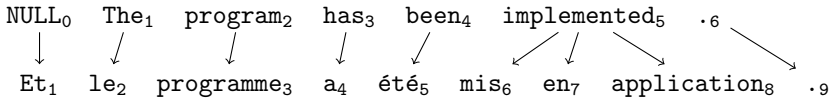


Figure 1: Example of word-alignment from English to French.

The probability of a translation, is the sum of the probabilities of all its possible alignments:

$$P(f|e) = \sum_a P(f, a|e) \tag{6}$$

The probability of an alignment depends on the IBM model that is used. In the simplest model (IBM 1), the probability of aligning two words f_j and e_{a_j} depends only on their translation probability $t(f_j|e_{a_j})$.

$$P(f, a|e) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(f_j|e_{a_j}) \tag{7}$$

Where l_e is the length of the English sentence e, and l_f the length of the French sentence f. The parameter ϵ is a normalization constant.

However, this model does not take into account the position of the words that are aligned. The first French word is as likely to be aligned with the first word of the English sentence, than with its last word. IBM Model 2 includes an alignment probability, which models the probability $P(j|i, l_f, l_e)$ of aligning an English word at position i with a French word at position j. In IBM 1 and 2, the alignment of a French word does not take into account the alignment of the other words in the sentence. All French words may be aligned to the same English word (and all other English words aligned to nothing). IBM Model 3 includes a fertility model, which models the number of French words each English word is aligned to. IBM Model 4 and 5 bring other improvements, such as improving the reordering model, by taking into account the movement of several consecutive words together.

The translation probabilities $t(f_j|e_{a_j})$ and alignment probabilities $P(j|i, l_f, l_e)$ are taken from probability tables that are built using the training data. However, the training data consists of sentence-aligned texts, which need to be aligned at the word-level in order to produce these tables. This is a *chicken-and-egg* problem, and it is solved by the *Expectation-Maximization* algorithm, which updates the parameters of the model iteratively, so as to maximize the alignment probability of the training data. Here is a quick outline of how the EM algorithm works:

1. Initialize the parameters of the model (e.g. with uniform probabilities);

2. Align the training data with the model;

3. Estimate model parameters from the aligned data;

4. Repeat steps 2 and 3 until convergence.

Initially, all alignments are as likely. But actual translations generally co-occur more often in the training data, than non-translations. For example, "kangaroo" and "kangourou" will probably be aligned more often than "squirrel" and "kangourou". Thus, at the parameters estimation step, the probability $t(kangourou|kangaroo)$ will be greater than $t(kangourou|squirrel)$, which will result in a better alignment at the next iteration.

Since the estimation of the model parameters is purely based on co-occurrence, the larger the parallel corpus used as training data, the better the translation model (greater word coverage, and more reliable alignment).

2.4 Phrase-based model

While the IBM models are still used for word-level alignment, word-based translation as been supplanted by other methods. Word-based translation has some shortcomings. In particular, it is unable to deal with *many-to-many* translations. For instance, the French phrase "casser sa pipe", would translate literally as "break one's pipe", but in most contexts the correct translation would be "buy the farm".

Phrase-based translation, by [Koehn et al., 2003], is the state-of-the art method in SMT. It is very similar to word-based translation, except that instead of independent words, small units of consecutive words, called *phrases* are translated. A phrase translation table is built by merging the word-based alignments in both directions. Table 1 shows examples of entries in a phrase translation table, for the French phrase "bien sûr".

Translation $\bar{e}_i$	Probability $\phi(\bar{f}_i\|\bar{e}_i)$
of course	0.5
naturally	0.3
of course ,	0.15
, of course ,	0.05

Table 1: Examples of phrase translation table entries for French phrase "bien sûr".

The first version of the model, by [Koehn et al., 2003], is defined as usual with a noisy channel model:

$$e = arg\max_{e \in e*} P(f|e)P(e) \tag{8}$$

$P(e)$ is a standard language model, augmented with a length factor to fix the bias of language models toward short translations. $P(f|e)$ is the phrase-based translation model. The English sentence e is decomposed into I phrases $\bar{e}_i$, which are translated into French phrases $\bar{f}_i$. The English phrases are reordered to form the output sequence e:

$$P(f|e) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)d(a_i - b_{i-1}) \tag{9}$$

$d(a_i - b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}$ (with $\alpha \leq 1$) is the distortion probability, which penalizes the phrase reorderings in the English sentence. a_i is the start position (in terms of words) of the phrase $\bar{e}_i$, and b_i the end position of $\bar{e}_{i-1}$. $\phi(\bar{f}|\bar{e})$ is the translation probability of phrase $\bar{e}$ into phrase $\bar{f}$, which comes from the phrase translation table.

State-of-the art phrase-based translation is implemented by the open source framework *Moses*, by [Koehn et al., 2007]. A log-linear model is used to include more features:

$$p(e|f) = \exp \sum_{i=1}^{n} \lambda_i h_i(e, a, f) \tag{10}$$

h_i are the features (e.g., language model, phrase model, distortion model, word penalty), and λ_i are their respective weights. The weights are usually automatically tuned, with a development corpus (their values are adjusted to maximize the translation quality of the sentences in the development corpus).

3 State of the art

Parallel corpora, which consist of sentence-aligned texts in different languages, are indispensable for statistical machine translation. These corpora are generally built from human translated texts, by aligning their sentences with automatic tools called *sentence aligners*. The result is a parallel text, a set of sentence pairs which are translations of each other. Commonly used parallel corpora include: the Europarl corpus, which assembles the proceedings of the European parliament (available in 21 EU languages), or the Canadian Hansards, which gather the official records of the Canadian parliament (in French and English).

However, such resources are limited in language coverage. While a lot of parallel resources are available for a common language pair like *French-English*, this is not the case for most language pairs.[1] Parallel resources also cover a limited range of domains. Most of these resources are made available by multilingual governmental institutions, and come from the political discourse domain. There is, for example, much less data covering everyday life spoken language.

To solve these problems, there has been a growing interest in using bilingual corpora that do not contain translated texts, but rather comparable texts, i.e., texts that share a common topic. Such bilingual corpora can be assembled from monolingual corpora (containing texts in a given language), which are much more common (e.g., Web pages, encyclopedias, newspapers, books, documentation). Several contributions have proposed methods to extract parallel sentences from such comparable corpora. In particular, an example of comparable corpus that has been thoroughly investigated, is the collaborative encyclopedia *Wikipedia*.

3.1 Sentence alignment

Sentence aligners take two texts which are translations of each other, and align them sentence by sentence, so as to produce a bitext, containing pairs of aligned sentences. This task is not easy, because these texts are not error free. It is very common that some sentences do not have a translation on the other side. Entire paragraphs might be missing. Moreover, a sentence does not

[1] Commercial machine translation systems, like *Google Translate* use English as a *pivot language* for most language pairs, e.g., for *Russian-German*, the text is first translated from Russian to English, and then from English to German.

necessarily translate to a single sentence. The punctuation might be different on the two sides, and a sentence on one side may correspond to two sentences on the other side.

Most sentence aligners are based on the assumption that the alignment is monotonous. If the positions of an English sentence e_i and its translation f_j are i and j. Then for all $k > i$, if f_l is the translation of e_k, then $l > j$. This assumption makes the alignment task much easier, by making possible the use of dynamic programming: the alignment at the positions i and j depends only on the alignment at positions $i - 1$, and $j - 1$.

There are two main types of sentence aligners: lexical-based aligners make use of lexical information to decide which alignment to do. They may use a bilingual dictionary or a translation system to translate one of the two sentences, and compare them. They may also use internal lexical evidence, like rare words, or cognates (words with the same root), or word co-occurrence. Length-based aligners, on the other hand, only use length information, i.e., they align together sentences whose length is similar.

Length-based alignment A very popular approach, that is length-based, is [Gale and Church, 1993]. It models the length distribution of the sentences, and uses dynamic programming to find the alignment which maximizes the overall score.

There are six kinds of possible alignments:

- *0-1* and *1-0* alignments, when a sentence is deleted or inserted (deletion or insertion).
- *1-1* are the most common alignments, where a sentence is aligned to exactly one sentence (substitution).
- *1-2* and *2-1*, where two sentences align with one sentence (expansion or contraction).
- *2-2* where two sentences align with two sentences (merger).

A dynamic programming algorithm decides at each step, which type of alignment is the most profitable, by using the following probability function:

$$P(match \mid \delta) \propto P(\delta \mid match) \times P(match) \tag{11}$$

$P(match)$ is the prior probability of a given type of alignment. Gale and Church estimated these probabilities from hand annotated data. For example, the most common alignment is *1-1*, and its probability is 0.89. $P(\delta \mid match)$ is estimated using a probability distribution on the length ratio of both sides.

$$P(\delta \mid match) = 2(1 - P(Z \leq |\delta|)) \tag{12}$$

The random variable Z follows a standard normal distribution, and $\delta = (l_2 - l_1 c)/\sqrt{l_1 s^2}$, where l_1 and l_2 are the numbers of characters on both sides, and the mean c and variance s^2 are the parameters of the model, which are determined empirically from the annotated data. This model is based on the assumption that the number of characters generated on the target side per character on the source side follows a normal distribution.

The dynamic programming framework computes for two given sequences of sentences $(s_1, \ldots, s_i)$, and $(t_1, \ldots, t_j)$, the minimum distance $D(i, j)$, while keeping track of the operations (deletion, insertion, substitution, etc.). The alignment corresponding to this distance is returned by the program.

This approach is mostly language-independent, since the model has only two parameters that need to be re-estimated for new language pairs, and their values are very similar for most pairs of

European languages. This method is also very fast. But, it is not perfect, and it can be misled by sentences with similar lengths, or by missing paragraphs. For this aligner to produce a good quality alignment, anchor points need to be defined, because the longer the texts to align, the more likely the alignment will fail at some point.

The well-known *Europarl* corpus, by [Koehn, 2005], has been aligned using the Gale and Church aligner. It is an example of an easy alignment, because the speakers turns (which person is currently speaking in the parliament) are specified using special markup, and they can be used as anchor points.

Hybrid The state-of-the-art aligners are both length-based and lexical-based. [Moore, 2002] uses a length-based method similar to [Gale and Church, 1993] to first align the corpus, and identify the sentences whose alignment has a high probability. Using these sentences, an IBM 1 word-alignment model is trained. A second pass combines the length-based probability and a translation probability, using this model.

YASA is a more recent sentence aligner, developed by [Lamraoui and Langlais, 2013]. Like [Moore, 2002], it uses both length-based information and lexical information. It uses *cognates* as lexical information. Cognates are words that have a common etymological origin. In this work, cognates are defined as words with a common prefix of at least four characters.

The limit of purely length-based aligners, like [Gale and Church, 1993], is that relying entirely on length evidence can lead to wrong alignments, in particular when entire paragraphs are missing. This can be improved, either by manually specifying anchor points (which limit the sizes of the portions of texts to be aligned), or by using lexical information to delimit the region where the alignment takes place.

In YASA, a simple dynamic programming technique is used, which aligns the two texts at the word-level, by using a score function which rewards cognate correspondences, and encourages to stick to the main diagonal. A sentence-level search space is then defined, by taking a fixed number of sentences centered around this alignment. Figure 2 illustrates this cognate-motivated search space reduction.

Finally, the sentence alignment is performed using a method similar to [Gale and Church, 1993], which combines the length-based score, with a cognate score. The search space is limited to the region defined in the previous step. YASA has been shown to be particularly robust to noise. Because it does not need to train a word-alignment model, it is also much faster than [Moore, 2002]. We will use YASA as a reference for sentence aligners in our work, as it shows state-of-the-art performance. The limit of YASA, is that it will work only with languages that share the same alphabet (European languages mostly).

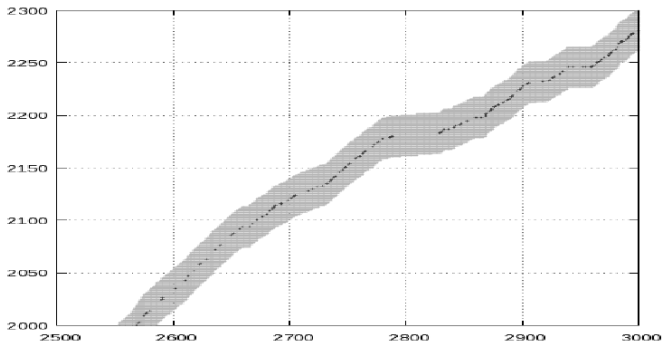


Figure 2: Word alignment identified from the cognates (dark line), and the delimited sentence-based search space (gray zone). The X-axis and Y-axis indicate French and English word positions in the VERNE bitext, from the BAF corpus.

3.2 Comparable corpora

Because of the scarcity of parallel resources, in most language pairs, and domains, a lot of work has been done toward the use of monolingual data in machine translation. Comparable corpora are sets of texts sharing similar topics. A popular example of comparable corpus is Wikipedia, which is a large source of data in many languages. Wikipedia articles in different languages on the same subject are linked with the so-called "interlanguage links".

The historical method for the extraction of parallel sentence pairs from comparable corpora is [Munteanu and Marcu, 2005]. It is applied on Arabic, Chinese and English news corpora. They pair together comparable articles, by using an IR engine. Parallel sentences are extracted from these article pairs, by generating all candidate pairs from the Cartesian product of the two documents. A first step filters out most of the unlikely sentence pairs. The remaining pairs are sent to a classifier, which decides whether or not they are parallel.

[Smith et al., 2010] use a similar method on Wikipedia, and extend it by adding features based on the Wikipedia markup (e.g., the hyperlinks that the sentences share). This work is applied on the English-Spanish, English-German and English-Bulgarian language pairs. Contrary to [Munteanu and Marcu, 2005], this method models the positions of the sentences to align. They observe that in Wikipedia, parallel sentences often appear in blocks (e.g. a translated paragraph). They use a CRF (Conditional Random Field), and features that take into account the position of the current sentences, and of the previously aligned ones.

[Rauf and Schwenk, 2011] propose an indexation-based technique. Source sentences are translated by means of a machine translation system, while sentences on the target side are indexed using an IR (Information Retrieval) engine. Translated source sentences are used as queries for the search engine. For each source sentence, the top 5 target sentences are returned. This generates candidate sentence pairs.

The measures *WER* (word error rate) and *TER* (translation error rate) are then used, between the target sentence, and the translated source sentence, to determine which of these pairs are parallel. Contrary to other methods that use a classifier, this method filters the sentence pairs by comparing their TER and WER against an empirically defined threshold.

They observe two kinds of errors produced by sentence alignment systems: either the two sentences share similar words but are not actually translations, or they are translations, but one of the two sentences has extra words at the end, that they call a *tail*. They propose a method for removing extra tails, that makes use of the WER.

3.3 Robustness of SMT to noise

While sentence aligners, and parallel sentences extraction methods try to achieve good alignment precision, they are never perfect. [Goutte et al., 2012] show that phrase-based statistical machine translation is very robust to noise in the training data. They first evaluate the amount of noise in parallel corpora that are commonly used for training SMT systems: Europarl, Canadian Hansards, United Nations, and Giga. Hansard has only 0.5% noise, and Europarl 1.2%. The UN proceedings have 2.8% noise, and the Giga corpus 13.1% noise. These corpora, in particular Giga (which is the largest one), are far from being error-free.

Then, they artificially introduce noise into sentence-aligned corpora (by permuting the sentences), and show that up to *30%* noise, the BLEU score of a phrase-based MT system trained with this data is mostly unaffected by the noise (the BLEU score drops from 37.59 to 37.31). In fact, up to 30% noise, removing the noisy sentences has a more negative effect on the language model (because it has less training data), than it helps the translation model, and it actually reduces the BLEU score.

SMT's robustness to noise can be explained by the random character of noise. The misalignments that are created because of the noise result in many wrong entries in the translation table. But these entries keep a low probability, and have little effect on translation (it is unlikely that the same misalignments occur many times).

This work suggests that in parallel sentence extraction methods, recall is more important than precision, provided that the precision stays above 60-70%. It would thus be prejudicial to enforce a high precision at the cost of a much lower recall.

4 Parallel sentence extraction system

We built a parallel sentence extraction system, that we will call *MUNT* in this report. This system is based on the work of [Munteanu and Marcu, 2005], and [Smith et al., 2010]. It takes as input a pair of documents, and returns all sentence pairs which are presumed parallel. Contrary to sentence aligners, it does not make any monotonicity assumption concerning the documents, thus every sentence pairs from the Cartesian product of the two documents are considered as likely to be parallel.

Munteanu & Marcu's system follows three main steps: article selection, candidate filtering, and parallel sentence selection. Figure 3 illustrates how these components are linked.

1. In the *article selection* step, similar articles which are likely to contain parallel sentences are paired.

2. The next step is the *candidate filtering*. The number of possible sentence pairs can be very large, and this steps eliminates most unlikely candidates.

3. The final step is the *parallel sentence selection*, which uses a classifier to select the parallel sentences from the remaining candidates.

The candidate articles to be paired come from two monolingual corpora. All three steps make use of a parallel corpus. This corpus is used to build a bilingual dictionary, for the article selection and candidate filtering steps. A parallel corpus is also used to train the classifier to recognize parallel sentences.

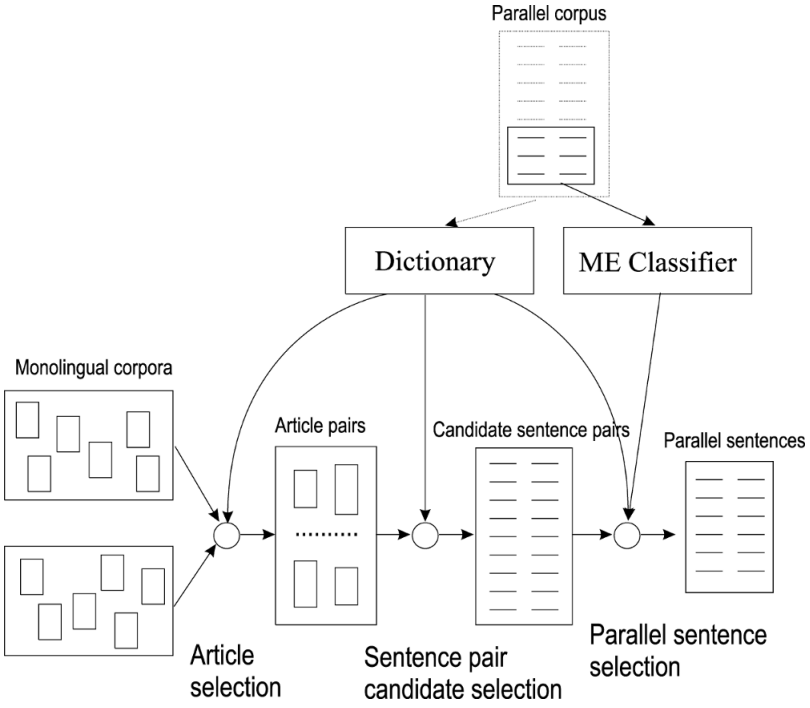


Figure 3: Munteanu and Marcu's parallel sentence extraction system

Our implementation, that we call MUNT, roughly follows Munteanu and Marcu's specification, with a few changes in the features that are used, and without the article selection step. Munteanu and Marcu use monolingual news corpora, and need a mechanism for pairing comparable articles together. We plan on using MUNT on Wikipedia, whose articles can be easily paired using the *interlanguage links* (which is an internal mechanism to Wikipedia, linking articles in different languages that are about the same topic).

4.1 Candidate filtering

Given a document pair, candidate sentence pairs are generated using the Cartesian product of the two sentence sets. However, as the Cartesian product can be very large, and most of these candidates are not parallel, a first step is needed to filter out the unlikely candidates in an efficient way.

The *word-overlap* filter, described by [Munteanu and Marcu, 2005], checks that both sentences share at least 50% of their words according to a bilingual lexicon. It also eliminate the sentence pairs whose length ratio (in number of words) is below a certain threshold. In our implementation, the word overlap is checked in only one direction. Checking both directions showed an important loss in recall with no improvement in precision. The ratio of source words that need to have a

translation on the target side depends on the parameter `min_overlap`, and the maximum length ratio is the parameter `length_ratio`. The default values for these two parameters are *0.5* and *2*. The bilingual lexicon is automatically built from a parallel corpus. The details of its construction are explained in the next subsection.

As Figure 4 shows, the word-overlap filter drops more than 98% of the candidates. This significantly reduces the class imbalance problem, which is a common problem in Machine Learning and can result in a low recall for the minority class.

Figure 5 shows the computation time, and precision and recall of MUNT with and without a filter. Despite a slight loss in recall (about 4 points), candidate filtering considerably reduces the execution time, and improves the precision. The default settings used in the experiments are detailed at the end of this section.

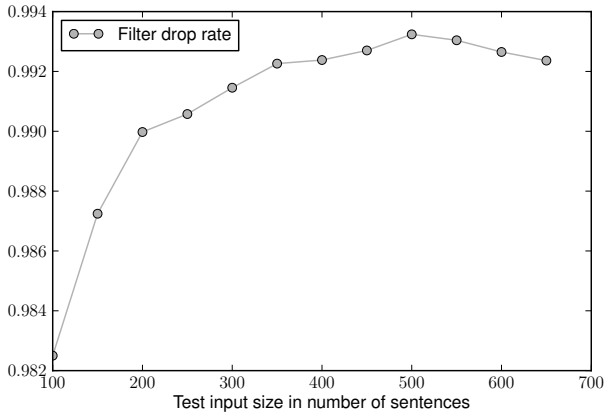
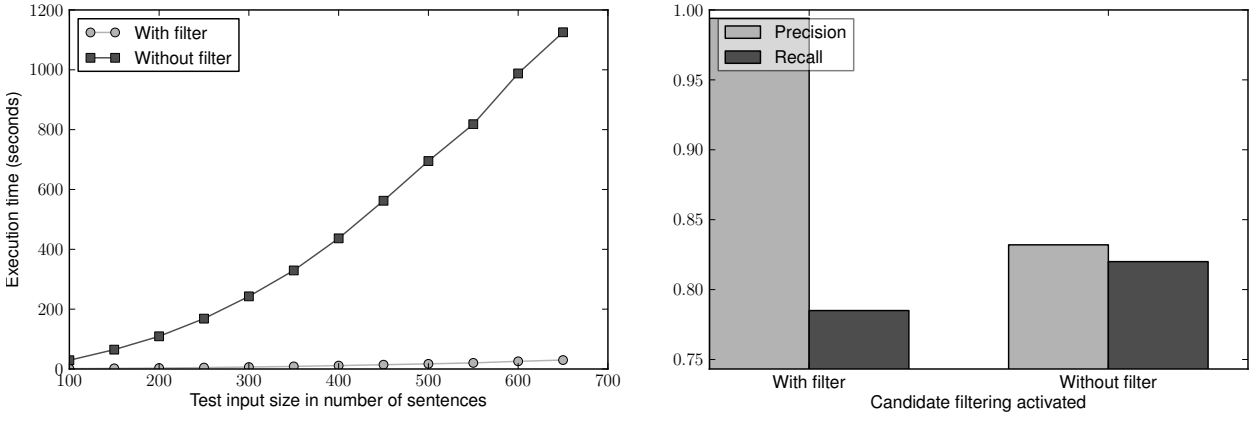


Figure 4: Drop rate of the word-overlap filter, depending on the size of the input.



(a) Impact of filtering on execution time

(b) Impact of filtering on precision and recall

Figure 5: Impact of the candidate filtering step on the performance of MUNT.

4.2 Word-level alignment

The features used by the classifier rely mostly on a *word-level alignment* of the two sentences. The alignment is performed using the *IBM 2* model, which takes into account the word translation probability and an alignment probability (depending on the length of the sentences and position of the words).

GIZA++ To build the alignment model, we use the software *GIZA++*, by [Och and Ney, 2003]. Examples of tables produced by GIZA++ are shown in Figures 6 and 8. Figure 7 shows an English vocabulary file, produced from a corpus using *plain2snt*.

One notable thing is that the most recent versions of GIZA++ have a bug, which result in the target sentence length column in the alignment table to always be 100. The correction for this bug proved to be remarkably easy. It is quite curious that this bug has never been fixed by the MT community, despite its severe consequences.

```
0 3 0.192065
0 4 0.0763278
0 6 9.67523e-06
0 8 0.0132089
0 11 0.015406
0 13 2.01435e-05
0 15 0.0526604
```

Figure 6: Examples of entries in a translation table produced by *GIZA++*. The first column is the index of the source word in the source vocabulary. Word indices start at 1, the 0 index is the `NULL` word. The next column is the index of the target word. The last column is the translation probability.

```
2 resumption 557
3 of 1697110
4 the 3600139
5 session 3079
6 I 532181
7 declare 1494
8 resumed 1342
9 European 274957
```

Figure 7: Examples of entries in an English vocabulary file produced by *plain2snt*, which contains all distinct tokens appearing in the corpus (including punctuation and numbers). The first column is the word's numerical index, which is used in the translation table. The third column is the number of occurrences of the word in the corpus.

```
1 1 1 1 1
0 1 2 2 0.0588196
1 1 2 2 0.882374
2 1 2 2 0.0588063
0 2 2 2 8.48772e-05
1 2 2 2 0.264736
2 2 2 2 0.735179
```

Figure 8: Examples of entries in an alignment table produced by *GIZA++*. The first two columns are the respective positions in the source and target sentence of the two words to be aligned. Word positions start at 1. The 0 position on the source side is used for the `NULL` word. The next two columns are the length of the source and target sentence in words. The last column is the alignment probability.

GIZA++ is able to produce word-level alignments using all five IBM models, and the HMM alignment model, by [Vogel et al., 1996]. To generate the translation and alignment tables, we use the IBM 3 and HMM models, which proved to produce tables of a better quality, and much lower size than IBM Model 2 (thus much faster to load into memory).

To train a GIZA++ model, the data needs to be tokenized (i.e., every words and punctuation separated by blank spaces), and truecased. The tokenization is necessary for GIZA++ to be able to distinguish tokens from each other, while truecasing reduces the vocabulary size by removing unnecessary uppercase letters (while proper nouns stay uppercased). For both steps, the *Moses* framework provides utility scripts: *tokenizer.perl* for tokenization, *train-truecaser.perl* for training a truecasing model, and *truecase.perl* for applying this model.

Word-alignment GIZA++ produces a word-level alignment of a given corpus, and creates an alignment model, but it does not provide a mechanism for using this model to align new sentence pairs.

IBM models 1 and 2 are relatively easy to use. There is actually no constraint on the number of words each target word can be aligned to, contrary to model 3 and higher, which include a target language fertility model. It is thus possible to determine the best alignment in a quadratic time, by simply choosing for each English word, the French word with the highest conditional probability.

Given an English word e_j, it is aligned to the French word at the position a_j, using the following equation:

$$a_j = arg \max_{i \in 0..l_f} t(e_j|f_i) \times a(i|j, l_f, l_e) \tag{13}$$

The index $i = 0$ on the French side corresponds to the `NULL` alignment. $t(f_i|e_j)$ is the probability of translating f_i to e_j, which is found in the translation table. $a(i|j, l_f, l_e)$ is the probability of translating a French word at position i into an English word at position j, where the length of the French sentence and English sentence are respectively l_f and l_e. This probability is found in the alignment table produced by GIZA++. We chose to ignore the target length l_e, because of the bug mentioned earlier. Even with a proper alignment table, taking it into account led to worse performance, because many tuples (i, j, l_f, l_e) had no entry in the table.

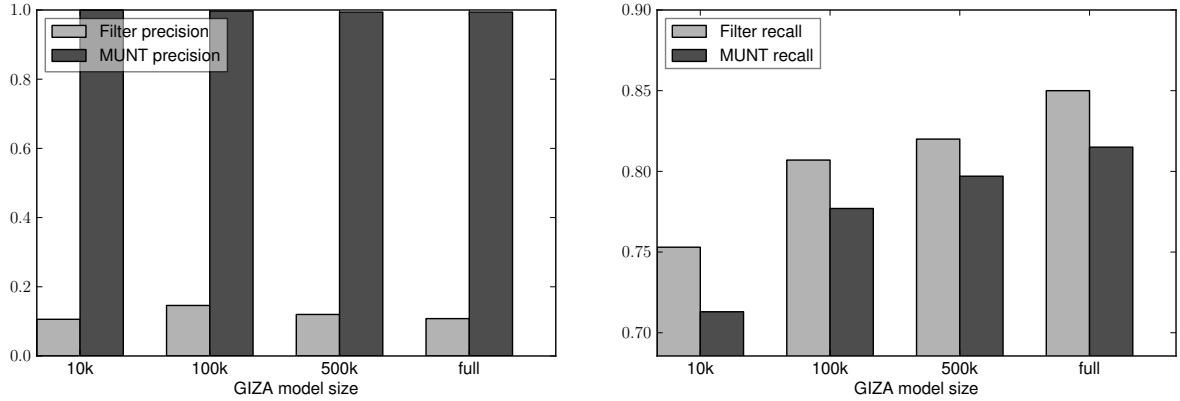
Since the alignment is a one-to-many relation, a straight-forward representation of an alignment is an array, where each position is an English word, and the value at that position is the index

of the French word it is aligned to. The representation of the alignment of Figure 1, would be `[0,1,2,3,4,4,5]`.

Lexicon The bilingual lexicon used by the word-overlap filter is extracted from the *GIZA++* translation table. To exclude the noisy entries, which do not correspond to actual translations, only the entries whose probability is over 0.1 are kept. This also helps reducing the size of the lexicon into memory. The corpus used for training the alignment model is *Europarl*. We tested with different corpora sizes (by taking the first n sentence pairs from Europarl): 10k, 100k, and 500k pairs, and the entire corpus (about 2M pairs).

Figure 9 shows the impact of the GIZA++ corpus size on the precision and recall of the filter, and on the final performance of MUNT.

The size of the GIZA++ model has a strong impact on the recall of the filter. This can be expected, as with a larger dictionary (with more entries, and more translations by entry), it is more easy to pass the word-overlap constraint. But the precision does no seem to be significantly impacted by the size of the dictionary, suggesting that, while the dictionary is larger, it is also more reliable. The default GIZA++ model that we use in our experiments is the *europarl_100k*.



(a) Precision (b) Recall

Figure 9: Impact of the size of the GIZA++ model on the precision and recall.

When parallel data is scarce (for example in language pairs with no parallel corpus available), MUNT can be bootstrapped using very few parallel data. Figure 9 shows that MUNT's precision is unaffected by the size of the parallel corpus. Even a corpus of only ten thousand parallel sentences shows a very good precision. A good method in this situation, is to use a small parallel corpus (for example, extracted from the web using a system like STRAND) to extract a larger parallel corpus from Wikipedia. This first pass does not produce the best recall, but using the extracted data as training data in a second pass would increase the recall.

4.3 Features and training

The features for a pair of sentences include features based on its word-level alignment in both directions, and length related features. We do not make use of Smith's Wikipedia features, because

we want our method to be applicable to other corpora than just Wikipedia. In particular, enormous monolingual *News* corpora are available, that would benefit from such a method.

The alignment features are the following:

- Number and percentage of source words which are connected;
- Number and percentage of source words which are not connected;
- The top three fertilities (i.e., number of words each source word is connected to);
- The percentage of source words with fertility 1, 2 and 3 or more;
- Length of the longest connected target substring;
- Length of the longest unconnected target substring;
- Log-probability of the alignment.

These features are computed for each alignment direction (from French to English, and English to French).

The general features are the following:

- Lengths of both sentences (in number of words), and their length ratio and length difference;
- Number of words shared by both sentences (e.g., numbers, or proper nouns);
- Percentage of source words that have a translation on the other side, for both directions (according to the lexicon used in the word-overlap filter).

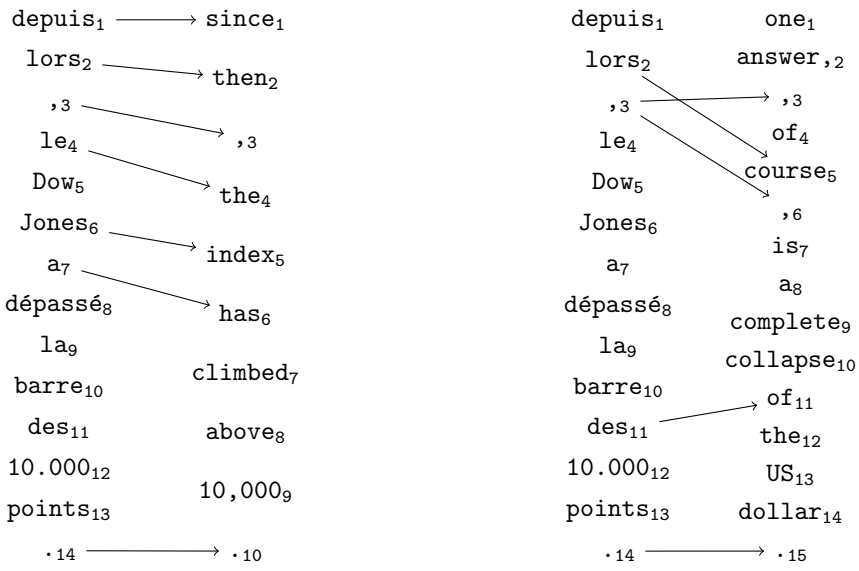
Figure 10 shows actual word-level alignments of sentence pairs, produced by the IBM Model 2. The first pair is parallel, while the second pair is not. The alignment is from French to English. French words can thus lead to several English words. English words that have no incoming arrow are actually connected to the French word `NULL`.

Alignments of parallel sentence pairs tend to have less crossings than those of non-parallel pairs, longer connected substrings, shorter unconnected substrings, a larger proportion of connected words, and their words tend to have a smaller fertility.

It can be observed that the first pair has the longest connected substring "since then , the index has". It also has a higher percentage of its words that are connected (7 out of 14 French words are connected, while only 4 are connected in the non-parallel pair). In the non-parallel pair, the comma on the French side is aligned to two commas on the English side. In this case, a higher fertility (a fertility of 2), shows a misalignment. Two commas are aligned to a single one, because there are no better candidates. The log probability of the parallel pair is also higher than the log probability of the non-parallel one (-85.6 against -158.1), showing that this alignment is more reliable.

These features are only computed for the sentence pairs which did not past the word-overlap filter. A Maximum Entropy classifier is trained using positive examples (real sentence pairs), and negative examples which pass the word-overlap filter. The positive examples are taken from a bitext. Negative examples are formed by pairing sentences at different positions in the bitext.

The MaxEnt classifier issues a probability for the two possible classes (`parallel` and `non-parallel`), that sum to one. We define a decision threshold s, such that a pair is considered as parallel if the probability of the class `parallel` is greater or equal than s.



(a) Parallel sentences

(b) Non-parallel sentences

Figure 10: Word-level alignments of a parallel sentence pair, and a non parallel sentence pair.

There remains the problem of multiple assignment: the system as it is can pair the same sentence twice. To ensure that sentences appear in at most one pair, we tested two different approaches. The first method is a greedy algorithm which orders the sentence pairs by best score, and iterates over this sequence by eliminating pairs whose source or target sentence were already seen in a previous pair (with a higher score).

Another method would be to choose the sentence pairs, so as to maximize the total score. Say (a, b) has a slightly higher probability than both (a, c) and (d, b), it could be a better strategy to choose the latter two, if they both have a high probability, contrary to what the previous solution would suggest. This problem is known as the *assignment problem*. The naive solution has a $O(n!)$ time complexity. The *Hungarian Algorithm* (or Munkres algorithm) solves this problem in polynomial time. Yet, we found this method to be too memory consuming and provided no advantage over the simpler greedy method.

Parameters There are several parameters which can impact the performance of MUNT. Table 2 lists all these parameters, and their default value. Unless specified otherwise, this value is the one that is used in all the experiments.

5 Experiments and results

The previous experiments showed the impact of some parameter values on the quality of the alignment. This section presents more thorough experiments, using different corpora, and different noise levels. Table 3 shows the different corpora that are used throughout our experiments, either for training or testing. The *europarl* corpora are used exclusively for training the GIZA++ models. The corpora used to train the MUNT classifier have the suffix *train* in their names.

Parameter name	Description	Default value
`giza_model`	GIZA++ model used for the word-alignment, and to build the lexicon.	*europarl_100k*
`lexicon_threshold`	Minimum probability for an entry of the translation table to be used in the lexicon.	0.1
`length_ratio`	Maximum length ratio used in the filter.	2
`min_overlap`	Minimum overlap used in the filter.	50%
`train_corpus`	Training corpus for the classifier.	*news.train*
`n_positive`	Number of positive examples used in the classifier training.	800
`n_negative`	Number of negative examples used in the classifier training.	800
`threshold`	Decision threshold of the classifier.	0.8
`test_corpus`	Evaluation corpus.	*news.test*
`n_test`	Number of lines used in the evaluation. The same number of sentences is taken from both texts (French and English)	400

Table 2: MUNT parameters, and default values.

Corpus	Domain	Number of sentence pairs
news.test	News	3000
news.train	News	2000
wiki.test	Wikipedia	1340
wiki.train	Wikipedia	1000
hans	Political	2856
verne	Literature	2276
citi1	Technology report	552
tao1	Science	353
europarl	Political	2,007,723
europarl_10k	Political	10,000
europarl_100k	Political	100,000
europarl_500k	Political	500,000

Table 3: Different corpora used in training and evaluation.

5.1 Experiment framework

On a computer programming point of view, we implemented MUNT using *Python 2*, which does not offer the best performance in terms of running time and memory consumption, but allows for quick prototyping. The library *Matplotlib* was used for drawing graphs and diagrams, and *NLTK*

for natural language processing (text processing tools and classifiers).

In order to easily run new experiments, we wrote a small experiment framework, which reads a configuration file with the chosen parameters. Figure 11 shows an example of configuration file. This system was inspired from *The Python Experiment Suite*.[2]

There are three kinds of parameters: those related to the initialization of the GIZA model (e.g., `giza_model_path`, `lexicon_threshold`), those affecting the training of MUNT's model (e.g. `n_positive`, `n_negative`, `min_overlap`), and those which only affect the evaluation itself (e.g., `n_test`, `noise_level`). The system allows the definition of default parameters (in the `DEFAULT` section), and possible overloading of these values in each experiment. If multiple experiments are defined, they are run in parallel. The `iterations` parameter causes the experiment to be run several times, and the results to be averaged. The results are saved in a file with the same name as the experiment, and the parameters of the experiments are saved in an additional file, with the extension `.cfg`.

Figure 11: Example of configuration file. The experiment "exp1" measures the precision and recall of MUNT, with a decision threshold ranging from 0.50 to 0.95.

```
[DEFAULT]
iterations = 10
path = ~/results
statistics = ['precision', 'recall']
giza_model_type = ibm2
giza_model_path = ~/model/europarl_100k
lexicon_threshold = 0.1
n_positive = 800
n_negative = 800
train_corpus = ~/data/news.train
length_ratio = 2
min_overlap = 0.5
n_test = 400
test_corpus = ~/baf/plain/citi1
threshold = 0.8

[exp1]
path = ~/my_experiment
threshold = arange(0.5, 1, 0.05)
```

Overfitting As our classifier includes more features than the original [Munteanu and Marcu, 2005] classifier, there is a concern that it could overfit to the training data, and be unable to generalize to new data. Table 4 shows that this does not seem to be the case. The classifier has a very similar precision and recall whether it is tested with the same data as was used in the training, or with a separate test set.

[2] http://www.rueckstiess.net/projects/expsuite

Training corpus	*news.train*	
Evaluation corpus	*news.train*	*news.test*
Precision	99.6%	99.4%
Recall	78.2%	76.9%

Table 4: Precision and recall of MUNT on the training data, and test data.

5.2 Noise generation

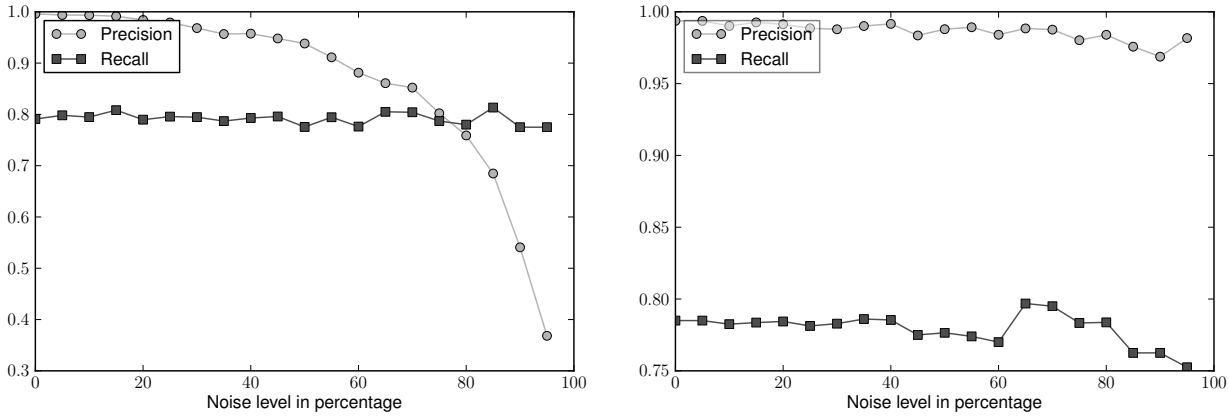
Previous experiments were done on bitexts. However, evaluation on bitexts is biased in a few ways, and probably does not reflect very well alignment performance on noisy data. Generating artificial noise on parallel corpora, can help evaluate how noise affects the performance of MUNT. We do not have access to sufficient amounts of noisy aligned data to perform such experiments on real data, and in addition, noise in real data is difficult to quantify, making evaluation difficult.

The noise generation method should reflect as well as possible the variations that can happen in noisy parallel corpora, while remaining simple and easily quantifiable. There are two methods that we know of in the literature: [Goutte et al., 2012] permute sentences in an aligned parallel corpus to show the robustness of statistical machine translation to noise in the training data. [Lamraoui and Langlais, 2013] remove sentences on the source side of a parallel corpus, to illustrate the robustness of the YASA aligner to noise.

While it is appropriate for SMT, Goutte's method is not adapted for evaluating sentence alignment methods. Indeed, mere sentence permutations is not what happens in noisy corpora, and parallel sentence extractors like MUNT, that do not make any monotonicity assumption can handle such changes easily (with MUNT, this has absolutely no effect, because the position in the corpus is not a selection criterion).

Lamraoui's method is more representative, sentence deletions on the source side equals to sentence insertion on the target side. However, since the deletions happen only on the source side, all remaining sentences on the source side remain aligned, making the task easier. The noise generated by this method has almost no effect on MUNT. Indeed, this causes more potential candidates for the target side of each alignment, but such alignment conflicts are easily handled by the greedy algorithm. On the other hand, MUNT is not as precise when dealing with sentence pairs in which both the source sentence and the target sentence do not belong to any true parallel pair, because it uses a global threshold (whose value is not tuned to fit exactly to the data). Relative comparison with other pairs is more precise than absolute comparison with an empirically defined threshold. An improved version of Goutte's noise, which consists of replacing a random number of sentences on the source side by external sentences, is more challenging to sentence alignment. Indeed both the source side and the target side have sentences that cannot be paired. Since the substitutions are done at random positions, this can cause small shifts. The impact of both noise generation methods on MUNT's precision and recall is illustrated in Figure 12.

We quantify the noise level with the simple formula: $noise = \frac{sentencesreplaced}{sourcesentences}$. Notice, however, that this method has its limits, as a sentence aligner like YASA will not be impacted the same way if entire blocks of sentences are replaced or if the substitutions are scattered across the corpus.



(a) Goutte noise (b) Lamraoui noise

Figure 12: Impact of noise on MUNT, using two different noise generation methods: Goutte and Lamraoui.

YASA evaluation Figure 13 shows the impact of artificially generated noise, using Goutte, on the precision and recall of both MUNT and YASA. Up to 40% noise, YASA keeps a higher recall than MUNT (by almost 20 points). YASA's precision is more sensitive to noise, but stays at a reasonable level until 30% noise. These results suggest that, up to 30% noise, using YASA would be beneficial. [Goutte et al., 2012], show that in parallel data for SMT, recall is more important than precision (provided that the precision stays above 70%).

It should be noted however, that in this experiment, the alignment reference consists only of one-to-one alignments (because it uses an already aligned bitext). This disfavors YASA, which is able to produce many-to-many alignments (if such an alignment happens, it is considered as wrong, even if partially correct).

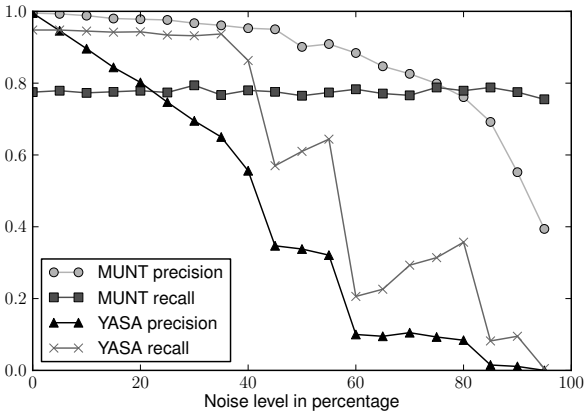


Figure 13: Impact of Goutte noise on MUNT and YASA.

5.3 Cross-domain experiments

[Munteanu and Marcu, 2005] show that MUNT achieves better results with *in-domain* data, than *out-of-domain* data. Indeed, if MUNT is trained with data from a certain domain, it will achieve better performance in this domain than in other domains.

Figure 14 shows MUNT's precision and recall with different test sets. *hans*, *citi1*, *tao1* and *verne* are extracted from the *BAF* corpus. The *wiki* corpus contains sentences from 100 different parallel Wikipedia articles (the method used to build this corpus is described in details in section 6).

The results are mostly consistent with Munteanu and Marcu's results: the domain has little effect on the precision, but strongly impacts the recall. This is surprising though, that the wiki test set achieves a significantly higher recall. There are two reasons that can explain this: the wiki set was formed using MUNT, which means that these pairs passed, and are thus less likely to be filtered out by mistake. The second reason, is that *wiki* is an easier corpus than *news*, because its sentences are of very dissimilar lengths (the standard deviation is 15.8 words on the English side of *wiki*, and 11.4 words in *news*).

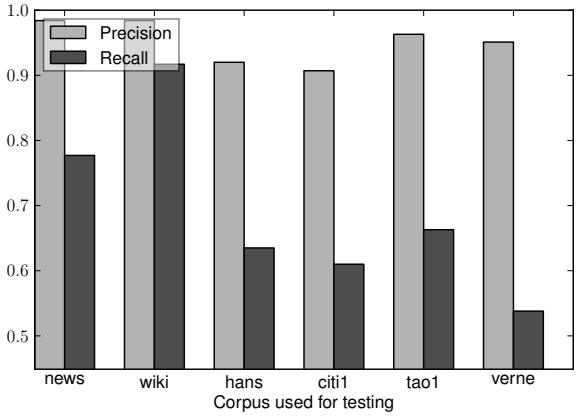


Figure 14: Impact of domain on MUNT's precision and recall.

Evaluation on real noisy data [Rebout, 2012] uses a set of manually annotated Wikipedia article pairs. It comprises 80 article pairs, of which 17 are considered as *quasi-parallel* (with a ratio of aligned sentences greater than 2/3). This corpus is downloadable on the website of the RALI.[3] After filtering out the articles with less than 10 sentences, and the articles with no alignment, the corpus contains 50 article pairs.

Figure 15 shows the results of both YASA and MUNT on the article pairs whose parallelism degree is above *0.3*. The parallelism degree, as defined by [Rebout, 2012] is:

$$score_{para} = \frac{2 * n_{para}}{n_{fr} + n_{en}} \tag{14}$$

Where n_{para} is the number of parallel pairs, and n_{fr} and n_{en} are respectively the number of sentences in the French and English articles.

[3] `http://rali.iro.umontreal.ca/rali/?q=fr/node/1293`

MUNT seems to handle the noise slightly better than YASA (in particular in terms of precision). This is difficult, however, to deduct anything from these results. This shows how difficult this is to quantify the noise in real world data. Had there been more data, we could have grouped articles of similar parallelism degrees, and averaged the precision on recall on those groups.

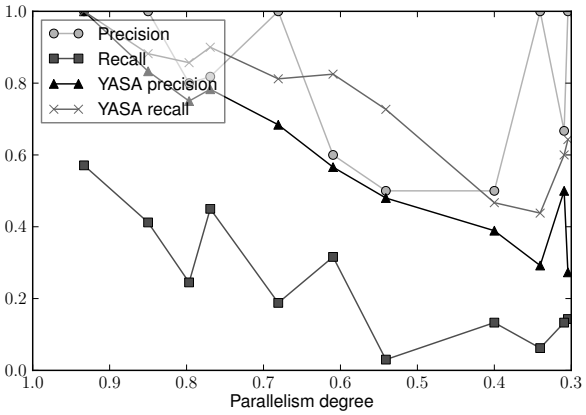


Figure 15: MUNT and YASA's precision and recall on Wikipedia article pairs.

6 Wikipedia

Wikipedia's collaborative aspect has made it the richest and largest encyclopedia in the world (it is currently the 6th most visited website on the planet). The English version of Wikipedia contains over 4.5 million articles, for a total of about 2.6 billion words, which is many times over the size of the second largest encyclopedia available in English. Its free access and electronic format make it an ideal candidate for natural language processing. But its most valuable advantage for machine translation, is that it is available in many languages. Wikipedia exists in 287 different languages. 49 of those languages have more than a 100,000 articles, and 123 languages have more than 10,000 articles. The *interlanguage links* connect all articles in different languages that discuss the same subject. For instance the French article "Kangourou" is linked to the English article "Kangaroo" and to the German article "Kängurus" (and to corresponding articles in 93 other languages). This feature is very useful to machine translation as it allows the pairing of related articles in different languages, making Wikipedia a very rich comparable corpus, in terms of size, domain coverage, and language coverage.

6.1 Article pairs extraction

Wikipedia's articles are rendered in HTML by the MediaWiki engine, which is run on Wikipedia's servers. Wikimedia, the organization responsible for hosting the Wikipedia project, shares up-to-date XML dumps of Wikipedia's articles for all languages, which are easily downloadable on the web.[4] Depending on the language, these XML files can be very large. For example, the compressed dump of all English Wikipedia pages is almost *20 Go*, which is roughly *100 Go* uncompressed.

[4] `http://dumps.wikimedia.org`

The XML tags delimit each article, and define their title, their index, and content. An article's content does not only consist of English text, but of *MediaWiki Markup*. Extracting the English text from a page is not an easy task. The markup uses keywords to specify the format of a page (e.g., headings, references, tables, hyperlinks), but it also contains templates making reference to other Wikipedia pages or to scripts, to render information in a unified way (e.g., dates, infoboxes, or tables). The only existing software program which is able to fully understand the wiki markup, is the MediaWiki parser.

It is possible to use the MediaWiki parser, thanks to the official web API. The API offers access to many features, such as the retrieving of articles by their title, or their index, or by content filtering. The property `extract` returns the content of an article in the HTML format. However, downloading hundreds of thousands of articles by this means could take weeks and would go against Wikipedia's policy (unless using a certified bot). That is why, using the official API is not a viable solution. We set ourselves to install the MediaWiki web server locally, in order to have a local access to the API. The default MediaWiki installation is not enough to have access to all Wikipedia's features, and a bunch of extensions need to be installed (including the Lua scripting feature, the API's extract feature). In addition, the *template* pages need to be indexed as well. Finally, we realized that the utilization of the API locally was even slower than using Wikipedia's servers (which are probably more powerful than our machine, and distributed). Thus, this latter, complicated solution is not a viable solution either.

We resigned to using *WikiExtractor*[5] to extract Wikipedia in the plain text format. It is a very simple parser, and it throws away important information (for example tables and lists), and is unable to perform template expansion (thus most dates are missing), but it works in a reasonable time.

We used the official API however, to retrieve the titles of the French and English articles that are linked by *interlanguage links*. These do not appear in the Wikipedia dumps anymore, as they belong to the *Wikidata* project. Wikidata centralizes interlanguage links from all Wikipedia projects for easier maintenance.

The following query retrieves the (potential) French article linked with the English article "Kangaroo": `http://en.wikipedia.org/w/api.php?action=query&prop=langlinks&titles=Kangaroo&lllang=fr`. The result is the XML in Figure 16, which gives the title of the corresponding article in the French Wikipedia.

Retrieving all the interlanguage links between the French and English Wikipedia can be done in a matter of hours using the API.[6] *919,001* title pairs were retrieved using this method.

Further processing was performed to segment the articles into sentences, and tokenize them into words. The result is a corpus named *wikipairs.tok*, which contains one sentence per line.

The sentence segmentation was done using *OpenNLP*. The Python library *NLTK* comprises such features (the function `sent_tokenize`), but there is no specialized sentence segmentation model for French. This results in poor sentence segmentation, where sentences are often cut in half because of unrecognized abbreviations or acronyms. This can be very problematic, since MUNT is unable to merge sentences. Like the GIZA++ corpus, the word tokenization was performed using Moses' tools (*tokenizer.perl*).

The titles of the articles are useful, because they are very often parallel. They were thus added

[5] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

[6] This is more efficient to use the French Wikipedia API, as there are three times less articles in the French Wikipedia.

```
<?xml version="1.0"?>
<api>
  <query>
    <pages>
      <page pageid="17064" ns="0" title="Kangaroo">
        <langlinks>
          <ll lang="fr" xml:space="preserve">Kangourou</ll>
        </langlinks>
      </page>
    </pages>
  </query>
</api>
```

Figure 16: Example of XML produced by Wikipedia's Web API.

to the content of the articles. For the articles we extracted, the average number of sentences per article is 32.0 in English, and 18.3 in French. The sentences contain in average 21 words in both languages. Table 5 gives more precise statistics concerning the number of tokens and sentences that were extracted.

Language	Total articles	Articles paired	Sentences (including titles)	Tokens
English	4,5M	919,000	29,371,994	630,882,642
French	1,5M	919,000	16,822,484	354,560,450

Table 5: Number of articles, sentences, and tokens extracted from Wikipedia, in English and French.

6.2 Execution of MUNT on Wikipedia

The `wikipairs.tok` corpus contains all pairs of articles, which are sentence segmented and tokenized. Articles are separated by `<article id="ID">` and `</article>` tags.

We separated this corpus into eight sub-corpora, in order to run MUNT on the eight cores of our machine. We kept the default parameters of MUNT: a decision threshold of 0.8, the giza model `europarl_100k`, and the training corpus `news.train`, with 800 positive examples, and 800 negative examples.

The execution took about 15 hours, and the result consists of *2.61 million* sentence pairs. A great number of these pairs are duplicates, in particular because of recurrent section headings (e.g. *Biography* and *Biographie*). When eliminating duplicates, the total is reduced to *2.26 million* sentence pairs. Keeping only the sentence pairs where both sentences contain at least 4 words, reduces the total to *1.92 million*.

The recall can be evaluated relatively to other methods, by counting the number of sentence pairs returned. To evaluate the precision, we randomly sampled *500* sentence pairs from the corpus, that we evaluated manually. We attributed three possible grades:

- A grade of *1* was attributed when the two sentences were found to be good translations, with no important missing information on either side.

- *2* means that one of the two sentences is a partial translation of the other.
- *3* means that the two sentences are not parallel at all.

Table 6 shows examples of sentence pairs returned by MUNT. We allow small missing parts in the *parallel* category. In the *partially parallel* category, one of the two sentences can miss an entire segment (examples 4 et 5). Examples 6, 7, and 8 show examples of misalignments. In example 6, both sentences are about chocolate temperature, with an obvious word-overlap, even though they are not translations. Short sentences like example 8, are a common case of error. The corpus used in training, *news.train*, does not have such short sentences. MUNT probably assumes that they are parallel, because they have the exact same length.

	Parallel (1)	
1	The Appalachian Trail is home to thousands of species of plants and animals , including 2,000 distinct rare , threatened , endangered , and sensitive plant and animal species .	Le sentier des Appalaches est le foyer de milliers d' espèces de plantes et d' animaux , dont distinctes rares , menacées ou sensibles .
2	Shortly after the signing of the decree , a military officer allegedly entered president Abdallah 's office and shot him , injuring Denard at the same time .	Quelques instants après la signature du décret , un officier des forces armées serait entré dans le bureau du président Abdallah et l' aurait abattu , blessant également Bob Denard .
3	Composition .	Composition .

	Partially parallel (2)	
4	At the cellular level , the nervous system is defined by the presence of a special type of cell , called the neuron , **also known as a "nerve cell" .**	À l'échelle cellulaire , le système nerveux est défini par la présence de cellules hautement spécialisées appelées neurones , **qui ont la capacité , très particulière , de véhiculer un signal électrochimique .**
5	In legends and oral traditions , the word distinguished ordinary mortal human beings —" tangata maori" —from deities and spirits **("wairua") ; likewise "wai maori" denoted "fresh water " as opposed to salt water .**	Dans les légendes et les traditions orales , le mot distingue les êtres humains mortels des dieux et des esprits .

	Non-parallel (3)	
6	After this point , any excessive heating of the chocolate will destroy the temper and this process will have to be repeated .	Cette décristallisation se produit lors de variations brutales de température ou lorsque le chocolat vieillit .
7	After his victory over Syphax , Masinissa commanded his skilled Numidian cavalry against the Romans in Spain , where he was involved in the Carthaginian victories of Castulo and Ilorca in 211 BC .	La langue punique fut d' usage courant dans sa capitale où l' on parlait également , en plus du berbère , les langues grecque et latine .
8	Health .	Politique .

Table 6: Examples of sentence pairs returned by MUNT.

The results of the manual evaluation are reported in Table 7. Only *48%* of the sentence pairs are found to be truly parallel. If we include the sentence pairs which are partially parallel, this makes *64%*. Even though statistical machine translation is very tolerant to noise, this is probably too much noise to be useful.

Grade	Number of pairs	Percentage	Cumulative
1 (parallel)	240	48%	**48%**
2 (partially parallel)	80	16%	**64%**
3 (non-parallel)	180	36%	100%

Table 7: Manual evaluation of 500 sentence pairs produced by MUNT (execution 1).

Intersection of YASA and MUNT YASA is a sentence aligner, and is normally used on parallel corpora. As the great majority of Wikipedia's articles are not parallel, YASA alone is not a viable option for building a bitext from Wikipedia. However, YASA might be used to align Wikipedia, and MUNT to check the returned alignments. Even though YASA produces a majority of wrong alignments, this technique considerably reduces the search space for MUNT.

Out of the *11 million* sentence pair candidates produced by YASA, *1.6 million* were kept by MUNT as parallel. *0.3 million* of those pairs are duplicates (frequent section headings). This technique, while it might seem a bit odd, is much faster than applying MUNT on the entire Wikipedia, and considerably improves the precision.

Table 6.2 shows the results of a manual evaluation on 500 sentences that were randomly sampled. The double-check improves the precision by 23 points compared to MUNT with the default settings. The estimated number of truly parallel sentences is about 1.0 million, while it was about 1.1 million using MUNT alone. The loss in recall is surprisingly low, if we consider YASA's inability to extract parallel sentences from non-parallel texts.

While YASA has a good recall with parallel documents, the alignments it produces on non-parallel documents are mostly rubbish. Sentence pairs that pass both YASA, and MUNT are thus likely to come from parallel, or partially parallel documents. It supports the idea that identifying these parallel documents, and processing them with YASA instead of MUNT could greatly improve the recall.

Grade	Number of pairs	Percentage	Cumulative
1 (parallel)	355	71%	**71%**
2 (partially parallel)	75	15%	**86%**
3 (non-parallel)	70	14%	100%

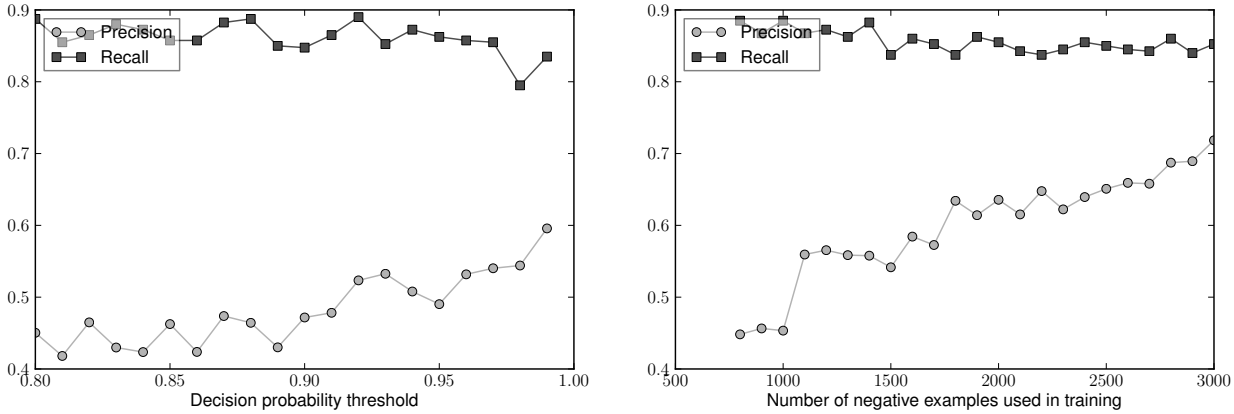
Table 8: Manual evaluation of 500 sentence pairs produced by the intersection of YASA and MUNT.

Creation of a Wikipedia training corpus As shown in the previous section, the domain of the training corpus has a strong impact on MUNT's performance. We collected sentence pairs from MUNT's results, in order to build a small Wikipedia corpus. The documents were ordered by score, using Rebout's score function: $score_{para} = \frac{2*n_{para}}{n_{fr}+n_{en}}$. n_{fr} and n_{en} are the respective lengths of the French and English articles in number of sentences. n_{para} is the number of sentence pairs returned by MUNT for this pair of documents. This simply counts the proportion of total sentences that belong to an aligned pair.

This would be difficult, using only this score function, to find out which documents are parallel, and which documents are not (because of MUNT's poor recall, a low score does not necessarily mean that the two documents are not parallel). However, the document pairs with the highest scores are likely to be parallel. We retrieved the 100 documents with the highest score, and manually checked

that they were parallel. We then used YASA on these documents, which can achieve a higher recall than MUNT on parallel documents, and obtained 2340 sentence pairs. This corpus was split in two corpora for training and testing purposes: *wiki.train* with 1000 sentence pairs, and *wiki.test* with 1340 sentence pairs.

Improvement of MUNT's settings Using the *wiki.train* and *wiki.test* corpora, we can tune the parameters of MUNT to improve its performance. [Munteanu and Marcu, 2005] train their classifier with as many positive examples as negative examples. However, when the two documents are not parallel, there is a strong class imbalance, and these settings can cause many false positive (hence a low precision). Figure 17 shows the positive impact of using more negative examples in training. The corpora used for training and testing are *wiki.train* and *wiki.test*. Noise is introduced artificially to a level of *90%* (which, we believe is representative of Wikipedia's average noise). Setting a higher decision threshold can also improve MUNT's precision. Surprisingly, both these settings have little impact on the recall.



(a) Impact of the decision threshold

(b) Impact of the number of negative examples

Figure 17: Impact of the decision threshold, and number of negative examples used in training on the precision and recall of MUNT, on a Wikipedia corpus with 90% noise. The default value for the decision threshold is 0.8, and the number or positive examples is 800.

Using a threshold of 0.98 and 3000 negative examples would probably result in a very good precision, but as demonstrated in [Goutte et al., 2012]'s work, recall is more important than precision. Another execution of MUNT on Wikipedia, by using the *wiki.train* corpus for training, with *2400* negative examples instead of *800*, and a decision threshold of *0.95*, resulted in a corpus of *1.87 million* unique sentence pairs. Table 9 shows the results of a manual evaluation on a subset of 500 sentence pairs. With an estimated precision of *84%*, and an improved recall (an estimation of *1.57M* parallel sentence pairs), these results are more convincing than those obtained with the default settings. The extracted parallel corpus, can be used for training a translation system.

6.3 Translation experiments and results

To evaluate the quality and usefulness of the sentence pairs extracted from Wikipedia, we trained a translation model using this data. Several parallel corpora are compared: the full Europarl

Grade	Number of pairs	Percentage	Cumulative
1 (parallel)	419	84%	**84%**
2 (partially parallel)	45	9%	**93%**
3 (non-parallel)	36	7%	100%

Table 9: Manual evaluation of 500 sentence pairs produced by MUNT (execution 2).

corpus, the Wikipedia corpus, and Europarl and Wikipedia merged together (Europarl+Wiki). As translation is very language dependent, the Europarl model is expected to produce better results on political test data, and Wikipedia on more general test data. Table 10 gives precise statistics about the size of the training corpora. Three parallel corpora from different domains are used in the evaluation: *hans* from the Canadian Hansards (political domain), *news.test* (news domain), and *wiki*, which was built with held-out sentence pairs from the Wikipedia corpus. Table 11 shows statistics about the test data.

The machine translation framework that is used in our experiments is *Moses*, by [Koehn et al., 2007]. It is a phrase-based SMT system.

The quality of a translation is evaluated using the BLEU metric (which compares the translation with the reference). BLEU has been shown to highly correlate with human evaluation. Though it is not perfect, and not as good as evaluation human, it is the most used automatic evaluation method in machine translation.

We trained a translation system for each of the training corpora, and report the BLEU scores obtained with each of the test corpora in Table 12.

The system trained with Wikipedia alone is not as good as the Europarl model, except with the in-domain *wiki* test set. However, merging Europarl and Wikipedia together improves the BLEU score of the *news* test corpus. Europarl performs better alone on the *hans* test set, which is political data, and in the same domain as Europarl. These results show that the parallel data extracted from Wikipedia, despite its non-perfect quality, can improve the translation quality for new domains.

Corpus	Counts	English	French
Europarl	sentences	1,964,110	1,964,110
	tokens	52,403,816	58,115,055
Wikipedia	sentences	1,849,226	1,849,226
	tokens	36,080,518	37,589,697
Europarl+Wiki	sentences	3,813,336	3,813,336
	tokens	88,484,334	95,704,752

Table 10: Training data size.

Corpus	Counts	English	French
hans	sentences	1,000	1,000
	tokens	22,583	25,059
wiki	sentences	1,000	1,000
	tokens	20,377	21,977
news	sentences	1,000	1,000
	tokens	26,939	34,909

Table 11: Test data size.

Training data	hans	wiki	news
Europarl (baseline)	**19.79**	22.90	23.68
Wikipedia	14.78 (-5.1)	33.65 (+10.8)	19.50 (-4.2)
Europarl+Wiki	17.81 (-2.0)	**34.26** (+12.4)	**25.29** (+1.6)

Table 12: BLEU scores under various conditions.

7 Parallel document extraction

Most comparable corpora, including Wikipedia include document pairs which are actually parallel. While a sentence extraction method like MUNT performs better than sentence aligners on non-parallel data, parallel texts remain the realm of the sentence aligners. A good approach, to improve the recall of parallel sentence extraction on comparable corpora, would be to identify these parallel documents, and process them separately with a sentence aligner.

7.1 Related work

In [Munteanu and Marcu, 2005], an indexation step is used to match *news* articles together. Documents in the source language are indexed using the *Lemur IR toolkit*. Documents in the target language are roughly translated into queries in the source language, by picking the 5 best translations for each word from a bilingual lexicon. Document pairs are then formed by taking the 20 top documents returned by the IR engine. However, this method is recall oriented, which means that most documents that are paired are only comparable, or not even comparable.

In [Rebout, 2012], a neural network is trained to find document pairs which are parallel, or nearly parallel. However, this method makes use of features which are specific to Wikipedia, such as the hyperlink sequences.

[Patry and Langlais, 2010] present PARADOCS, a system for pairing parallel documents together. This system uses an indexation based method for reducing the search space, and then a classifier which uses features based on hapaxes (rare words), numbers and punctuation. Yet, this system is intended to be used on a large collection, for finding likely parallel document pairs. Experiments show poor results on Wikipedia: only *58%* of the sentence pairs returned are actually mostly parallel. The results are interesting, because they show on a small subset of 50 article pairs that 26% of the articles paired by interlanguage links are parallel or mostly parallel.

Other systems, like STRAND by [Resnik and Smith, 2003], are specific to parallel document extraction from the web. STRAND finds candidate document pairs on a given Web domain by looking for patterns in the URLs, like `/en` or `/fr`. The HTML structure of the web pages is then flattened, and compared using an edition distance. This works on web pages, because translated pages often exhibit a very similar HTML structure (actually the same template is often used, in which text is substituted). In the general case, however, (in Wikipedia, or News corpora) there is no structural evidence that two documents are parallel.

7.2 Document selection system

We have observed that YASA has a higher recall than MUNT, when the documents are mostly parallel. It would thus be beneficial to find the document pairs which are parallel in Wikipedia, and align them with YASA. The number of parallel sentences obtained with the intersection of YASA and MUNT's results suggests that a significant number of document pairs are parallel, or parallel enough for YASA.

We build a Maximum Entropy classifier, with two classes: *parallel* and *non-parallel*. It uses document level features, based on their length (in number of sentences, or words), or the alignment results produced by YASA and MUNT. More specifically, the features are the following:

- Length ratio and length difference in number of sentences;
- Length ratio and length difference in total number of words;
- Proportion of YASA's alignment which are null alignments;
- Proportion of sentences which belong to a null alignment (ratio of sentences which are not paired);
- Number of pairs returned by MUNT;
- Number of pairs returned by MUNT over the total number of sentences;

Training data The corpus used by [Rebout, 2012] is not large enough for our purposes. It contains only 80 article pairs, and many of those are very short. The number of parallel article pairs is only 17. Moreover, the extraction technique that was used is different from ours (the articles still contain some wiki markup), and so is the sentence segmentation and tokenization.

A manual annotation of articles was not a viable option in our case. It would require reading hundreds of articles in order to acquire enough positive examples, which is not feasible (nor desirable) in our limited time. We select as positive examples, the 100 article pairs that have the greatest ratio of aligned sentences, according to MUNT's results in the last section. Three hundred article pairs are randomly taken, among all articles whose ratio is under 0.5. This corpus is far from ideal, because the positive examples are the most parallel articles, that MUNT was able to identify. The classifier will probably have trouble to generalize to all *mostly* parallel articles.

Evaluation Since we have very few data, we use cross-validation, in ten folds. Cross-validation works by running the experiment as many times as there are folds. Each time a different portion of the data is used for the evaluation (10% of the total in our case), and the rest for training. The final results are the average of all individual experiment.

Table 13 shows the confusion matrix obtained with this evaluation. The classifier achieves a 99% precision on our test data, with a 80% recall. This is very good, compared to the results obtained by [Patry and Langlais, 2010].

	Predicted parallel	Predicted non-parallel	Total
Parallel	80	20	100
Non-parallel	1	299	300

Table 13: Evaluation of the parallel document extraction method.

We observed that the occasional false positive which occurred were actually very short articles. We extracted article pairs from Wikipedia, by using this method, with an arbitrary decision threshold of *0.99*, and eliminating the articles with less than 5 sentences. This resulted in 39,284 article pairs, with a total of 564,588 sentences in English, and 560,504 sentences in French. Aligning these articles with YASA, produced a total of 497,913 sentence pairs. The total number of sentence pairs produced by MUNT for these articles was 217,000.

Even though the recall seems to have improved, a manual evaluation of 100 randomly sampled sentence pairs, showed a precision of only 32% (with grade 1), 44% when counting the partially parallel pairs (grade 2). This is significantly lower than the precision achieved with MUNT (84% with grade 1). If we estimate the total number of correct pairs, our method, which is supposed to improve the recall, actually results in an inferior recall (160k sentence pairs instead of 180k).

These results are much below our expectation (and very different from those obtained with the previous evaluation). This can be explained by the lack of good quality training data. The training data used here is not very representative of the entire Wikipedia collection, as the article pairs that were chosen as positive examples are the *most parallel* pairs. There is also probably too few data, considering the variability of this data (e.g., article length variability). In particular, in our training data, very few negative article pairs are of similar length. Moreover, the settings, in particular the balance of positive and negative examples are probably ill-adjusted to the real data.

As the results remain inconclusive, we did not try to train a machine translation system with this data.

8 Conclusion

We have implemented [Munteanu and Marcu, 2005]'s method for extracting parallel sentences from comparable corpora. Our main contribution is a more in-depth evaluation of this system, which measures the impact of several parameters, including the word-overlap filter, the domain of the training data, the value of the decision threshold, and the ratio of positive and negative examples used in the classifier training. Furthermore, we introduce artificial noise into a parallel corpus, to measure how MUNT's performance degrades when applied to non-parallel corpora. Even though this system was designed for comparable non-parallel corpora, its performance is indeed impacted by the parallelism degree of the input corpus. Depending on the corpus, a more precise tuning of the parameters can be required, in order to achieve a reasonable precision.

We apply the MUNT system on a collection of article pairs extracted from the famous Wikipedia encyclopedia. A parallel corpus of about 1.9 million pairs is thus created. By the means of a BLEU evaluation on translation systems, we show that merging this corpus with the Europarl corpus can improve translation performance.

We also conduct experiments on the YASA sentence aligner, to compare its robustness to corpus noise with MUNT. The results of our experiments show that sentence aligners, if the input corpus is parallel enough, can bring a better recall than a parallel sentence extraction system like MUNT. This led us to believe, that under the rich variety of parallelism levels in the Wikipedia corpus (some documents are actually parallel), extracting the documents which are mostly parallel, and processing them separately from the rest with a sentence aligner like YASA, might help us collect more parallel sentence pairs from Wikipedia.

Our final contribution is a method to be applied on document pairs, which decides whether or not they are parallel. The results for this last part are inconclusive, because of a lack of suitable training resources.

Our future work (in the coming weeks) will be to improve this parallel document extraction system, by finding more representative training data (and in larger quantities), adding more features, and tuning the settings to better fit to the problem at hand. We will then show that this method, when applied on Wikipedia, can improve the recall of parallel sentence extraction.

We will also make our source code and resources available online, for the use of people who are interested in a ready to use parallel sentence extraction system.

References

[Brown et al., 1993] Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311.

[Fung and Cheung, 2004] Fung, P. and Cheung, P. (2004). Multi-level bootstrapping for extracting parallel sentences from a quasicomparable corpus. In *Proceedings of COLING 2004*.

[Gale and Church, 1993] Gale, W. A. and Church, K. W. (1993). A program for aligning sentences in bilingual corpora. *Comput. Linguist.*

[Goutte et al., 2012] Goutte, C., Carpuat, M., and Foster, G. (2012). The impact of sentence alignment errors on phrase-based machine translation performance. In *AMTA-2012: the Tenth Biennial Conference of the Association for Machine Translation in the Americas*.

[Koehn, 2005] Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54.

[Lamraoui and Langlais, 2013] Lamraoui, F. and Langlais, P. (2013). Yet another fast, robust and open source sentence aligner. time to reconsider sentence alignment? In *XIV Machine Translation Summit*.

[Langlais et al., 1998] Langlais, P., Simard, M., and Véronis, J. (1998). Methods and practical issues in evaluating alignment techniques. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 711–717.

[Moore, 2002] Moore, R. C. (2002). Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, AMTA '02, pages 135–144, London, UK, UK. Springer-Verlag.

[Munteanu and Marcu, 2005] Munteanu, D. S. and Marcu, D. (2005). Improving machine translation performance by exploiting non-parallel corpora. *Computat. Linguist.*, 31:477–504.

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

[Patry and Langlais, 2010] Patry, A. and Langlais, P. (2010). Paradocs : l'entremetteur de documents parallèles indépendant de la langue. *Traitement Automatique des Langues*, 51.

[Rauf and Schwenk, 2011] Rauf, S. A. and Schwenk, H. (2011). Parallel sentence generation from comparable corpora for improved smt. *Machine translation*, 25(4):341–375.

[Rebout, 2012] Rebout, L. (2012). L'extraction de phrases en relation de traduction dans wikipedia. Master's thesis, Université de Montréal.

[Resnik and Smith, 2003] Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *Comput. Linguist.*, 29(3):349–380.

[Smith et al., 2010] Smith, J. R., Quirk, C., and Toutanova, K. (2010). Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 403–411.

[Vogel et al., 1996] Vogel, S., Ney, H., and Tillmann, C. (1996). Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA. Association for Computational Linguistics.