Statistical NLP - Assignment 4
Word Alignments

Daniel Rivera Ruiz
Department of Computer Science
New York University
New York, NY 10003
drr342@nyu.edu

1 Problem Definition

The current assignment deals with the problem of word alignments defined as follows: given a source sentence in English $e_1 e_2 ... e_l$ and its translation into French $f_1 f_2 ... f_m$, find alignment variables $a_1 a_2 ... a_m$ where each of the a_i can take a value $j \in \{0...l\}$, meaning that the i^{th} word in the French sentence is aligned (translates) to the j^{th} word in the English sentence. Notice that j can take the value zero because a `NULL` word is introduced at the beginning of the English sentence to align French words that have no counterparts in English.

Three algorithms were implemented for this assignment: a heuristic algorithm described in section 2, and the IBM Models 1 and 2 described in section 3. All the algorithms were trained and tested with data extracted from the Hansard Corpus, which consists of parallel texts in English and Canadian French drawn from official records of the proceedings of the Canadian Parliament. To evaluate and compare the algorithms, the alignment error rate (AER) metric as it is defined in Parikh (2017) was considered throughout the assignment. Other metrics like precision (P) and recall (R) are presented in section 5 along with the complete results and conclusions of the experiments conducted.

2 Heuristic Approach

As described in the assignment definition (Parikh, 2017), a heuristic approach to word alignment considers only simple statistics taken directly from the training corpus to match up words based on some measure of association. In this case, a score $S = \frac{c(f,e)}{c(e) \cdot c(f)}$ is assigned to each observed pair of words (f, e) during training time . At test time, any given French word is aligned to the English word that maximizes the score $S(f, e)$ or to `NULL` if the pair (f, e) was never seen during training. Using 10,000 sentences from the Hansard Corpus (in addition to the 37 that were originally considered for the baseline model) the heuristic algorithm achieved an AER of 52.50%.

3 IBM Models

The IBM models were conceived to address the problem of automated machine translation, where the objective is to propose a sentence e in a source language E (English) as the translation of a given sentence f in a foreign language F (French): $e = \text{argmax}_{e \in E} P(e|f)$. To achieve this, the IBM models use the noisy-channel approach to introduce a translation model $P(f|e)$ and a language model $P(e)$.

To facilitate its modeling, $P(f|e)$ is rewritten in terms of the alignment variables a_i and parametrized as $P(f_1...f_m, a_1...a_m|e_1...e_l, m) = \prod_{i=1}^{m} q(a_i = j|i, l, m)t(f_i|e_{a_i})$, where t can be interpreted as the conditional probability of generating French word f from English word e, and q refers to the probability of alignment variable a_i taking the value $j \in \{0...l\}$, conditioned on the lengths l and m of the English and French sentences (Collins, 2011).

The problem with this approach is that the alignment variables are usually not available as part of the training examples. Given this condition of partially observed data, the parameter estimation for the IBM models becomes an *Expectation-Maximization* (EM) problem. The intuition behind EM is the following:

1. Initialize the model parameters t and q to some (random) values.
2. **E step:** Compile the *estimated counts* $c(f,e)$, $c(e)$, $c(j|i,l,m)$ and $c(i,l,m)$ using t and q.
3. **M step:** Re-estimate the parameters t and q with the counts calculated in step 2 and iterate.

The specifics as to how the parameters are re-estimated and the pseudo-code for the complete algorithm can be found in Collins (2011). After executing the EM algorithm, the parameters of the model will converge (up to a certain precision depending on the number of iterations) to a local optimum. Convergence to the global optimum cannot be guaranteed because the log-likelihood function being optimized by the algorithm is non-convex.

Finally, using the model parameters t and q the alignment variables can be retrieved with the following expressions:

$$\begin{aligned} a &= \underset{a_1 \ldots a_m}{\operatorname{argmax}} P(a_1 \ldots a_m | f_1 \ldots f_m, e_1 \ldots e_l, m) \\ a_i &= \underset{j \in \{0 \ldots l\}}{\operatorname{argmax}} q(j|i,l,m) t(f_i|e_j) \end{aligned} \tag{1}$$

3.1 IBM Model 1

In the general definition of the IBM models described above, the parameter q is associated to a probability distribution over the alignment variables. In the particular case of Model 1, the assumption is made that for all values of i, j, l and m, $q(j|i,l,m) = \frac{1}{l+1}$. This condition means that all the alignments are considered to be equally likely and considerably simplifies the model.

The algorithm for IBM Model 1 as it is described in Collins (2011) was implemented to estimate the parameters $t(f|e)$. For the implementation, the number of iterations of the EM algorithm was set to 20 and the size of the training data set was gradually increased to improve the overall performance of the algorithm.

To retrieve the alignments, equation 1 is reduced to selecting the argument that maximizes the parameter $t(f|e)$. At test time, French words that were never seen during training or that were never aligned to any of the words in the associated English sentence were aligned by default to `NULL`. With these configurations, and using the additional 10,000 sentence pairs from the Hansard Corpus during training, IBM Model 1 produces an AER of 42.25%.

3.2 IBM Model 2

The main difference between IBM models 1 and 2 is that in model 2 no assumptions are made about the q parameters, and therefore it will be up to the model to learn them along with the t parameters. Another important difference comes at the moment of initialization. While model 1 just used random values, model 2 will take advantage of the t values yielded by model 1 and use them as a first approximation.

Under these conditions, the final implementation for model 2 is described in the following steps. This implementation is also based in the notes from Collins (2011), where the complete derivations and equations can be found.

1. Run 20 iterations of the IBM Model 1 algorithm to estimate initial t parameters.
2. Initialize q parameters to random values.
3. Run 20 iterations of the EM algorithm for Model 2, updating t and q after each iteration.

Finally, equation 1 is used to predict the alignment variables using the values generated for the parameters t and q. Just as with model 1, French words that were never seen during training or that were never aligned to any of the possible English words were aligned to the `NULL` word at test time.

Using the additional 10,000 sentence pairs from the Hansard Corpus, IBM Model 2 achieves an AER of 33.93%.

Given that the objective of the assignment was to achieve an AER of 17.3% on the test set, the model had to be further modified (or a new model had to be proposed) in order to get a score that was closer to this threshold. The improvements that were implemented to achieve this objective are described in the following section.

4 Boosting the Performance

While experimenting with the three models described in sections 2 and 3, it became clear that incrementing the size of the training data set resulted in an improvement of the performance. With this intuition in mind, the first approximation to boosting the performance of IBM Model 2 was simply to introduce more sentence pairs to the training set.

The results obtained were quite positive: with a training data set of 100,000 sentence pairs the AER went down almost ten points for a final value of 24.18%. By further increasing the size of the training data set in one order of magnitude (for a total of 1 million sentence pairs) the AER reached its lowest value yet at 18.90%.

At this point, the model was already yielding results relatively close to the 17.13% target value, and so the straightforward thing to do would have been to further increase the size of the training data set. However, the number of sentence pairs available for training wasn't much bigger than 1 million, so just including them all was unlikely to improve the performance of the model substantially.

The next step taken originated from the thought that the testing being might have been extracted from a different corpus as the training set. If that were the case, the correlation between the two sets wouldn't be very strong and the model could be overfitting the training data. To discard this possibility, additional data from a different corpus were introduced to the training set.

Europarl is a parallel corpus designed for statistical MT applications introduced by Koehn (2005). It was extracted from the proceedings of the European Parliament and includes versions in 21 European languages. For obvious reasons, only the English-French sentence pairs were considered for the task at hand.

In order to maintain consistency among the data, the sentences extracted from Europarl were preprocessed to match the formatting of the Hansard Corpus. The preprocessing consisted mainly of changing the file encoding, inserting blank spaces to separate punctuation as individual tokens, and eliminating contractions in the French sentences (e.g. *du* ⇒ *de le*). The final version of the Europarl English-French sentence pairs that was used for this assignment can be retrieved here.

For each of the boosting experiments developed with data from the Hansar Corpus alone (10k, 100k, 1M), equivalent experiments were designed using twice the data where the additional sentence pairs were extracted from the Europarl corpus. Including these sentences improved the performance of the model in about 3 points for all the experiments, resulting in a final AER value of 15.93% with the 2M training data set.

5 Results and Conclusions

Table 1: Performance comparison of the Word Alignment Algorithms implemented in the assignment. All algorithms were trained with approximately 10,000 sentence pairs extracted from the Hansard Corpus.

Algorithm	Precision	Recall	AER
Heuristic	0.4563	0.5148	0.5250
IBM Model 1	0.5208	0.6982	0.4225
IBM Model 2	0.6332	0.7101	0.3393

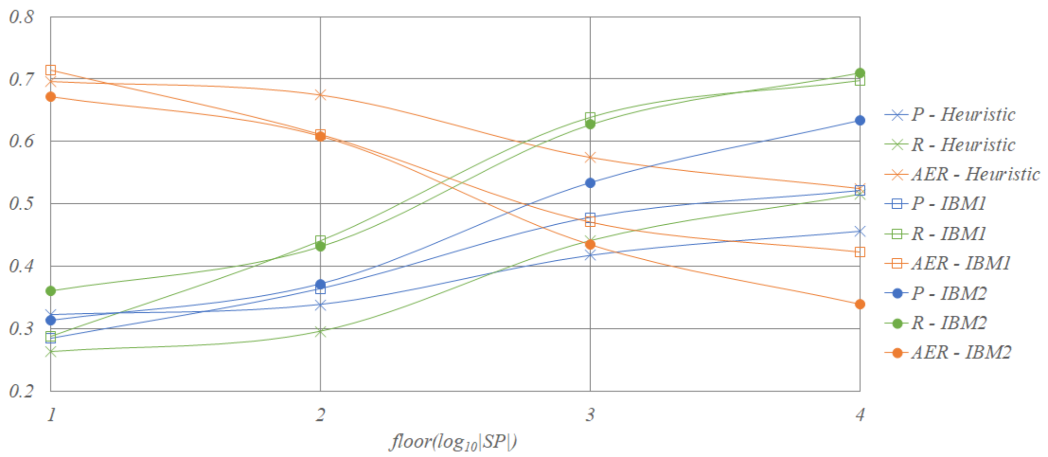

Figure 1: AER values for IBM Model 2 with different sizes $|SP|$ of the training data set.

Table 1 and Figure 1 show the results for the three original experiments conducted throughout the assignment, including scores for precision, recall and AER. As it was expected, IBM Model 2 yielded the best results for all three metrics.

Finally Table 2 presents the results for section 4, where the number of sentence pairs was gradually incremented and an additional source of data was introduced to boost the performance of IBM Model 2. It can be observed that as the size of the training data set increases, so does the overall accuracy of the algorithm in terms of the AER metric.

The most important conclusion that can be drawn from this observation is that the size of the training data set plays a very important roll in learning problems like the one considered throughout this assignment. With the ever-growing amount of data available online and the computational power of modern computers it is possible to considerably boost the performance of any algorithm regardless of its complexity.

Table 2: Results obtained with IBM Model 2 after incrementing the size of the training data set and incorporating data from the Europarl corpus.

Corpus		Hansard			Hansard + Europarl	
Sentence Pairs	10k	100k	1M	20k	200k	2M
Precision	0.6332	0.7431	0.8183	0.6783	0.7813	**0.8643**
Recall	0.7101	0.7840	0.7988	0.7337	0.7959	**0.8047**
AER	0.3393	0.2418	0.1890	0.3012	0.2133	**0.1593**

References

Michael Collins. Statistical Machine Translation: IBM Models 1 and 2, 2011. URL `http://www.cs.columbia.edu/~mcollins/ibm12.pdf`.

Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. *MT Summit 2005*, 2005. URL `http://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf`.

Ankur Parikh. Statistical NLP: Assignment 4, 2017. URL `http://www.cs.nyu.edu/~aparikh/assignment4_fall2017.pdf`.