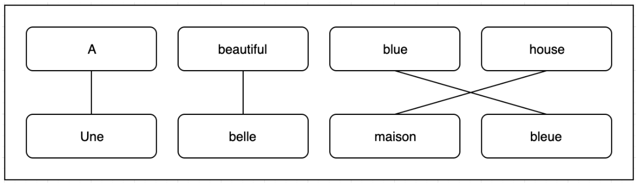
Word Alignment Models for Machine Translation Task Algorithms for NLP 11-711: Assignment 4

Kinjal Jain[1]

Abstract— **This paper addresses various models employed for generating word alignments given a parallel corpus of French and English sentences. Some heuristic approaches along with IBM Model 1 and HMM based Model are implemented and evaluated extrinsically based on AER and BLEU score measures from the machine translation task. The different models implemented are constrained by memory usage, train time and decoding speed for the translations.**

I. INTRODUCTION

Word alignment is one of the key Natural Language Processing tasks, used for identifying relationships among the words in a bitext (text with two parallel language sentences). Word Alignments are used to generate phrase tables which are used in various tasks like Statistical Machine Translation. For example, below is the picture of a possible word alignment between a French and an English sentence denoted by the line joining the words.



There are multiple ways to approach the task of finding good word alignments, but this paper specifically discusses two algorithms namely IBM Model 1 and Hidden Markov Model based Alignment Model. Following section covers them in detail along with some heuristic based approaches.

II. ALGORITHMS AND IMPLEMENTATION DETAILS

For the following models discussed, French and English language pairs are used.

Heuristic Model: As part of this model two approaches were experimented. Let f denote a word in French and e denote a word in English. Then, $c(f)$ denotes the number of sentences in training data where f occurs, and $c(e)$ denotes the number of sentences in training data where e occurs. And, $c(f,e)$ denotes the number of pair of French, English sentences where f occurs in French sentence and e occurs in the English sentence. Then the following approaches were tried to compute the alignment score and given the pair of sentences, the f and e with maximum score were aligned together:

[1] Email:kinjalj@andrew.cmu.edu

1. Pointwise Mutual Information Score: $\frac{c(f,e)}{c(f)c(e)}$
2. Dice Coefficient Score: $2 * \frac{c(f,e)}{c(f)+c(e)}$
3. Ochichai Score: $\frac{c(f,e)}{\sqrt{c(f)+c(e)}}$

$HeuristicAligner$ class implements the heursitics based models. $IntCounter$ is used for storing the counts of each French and English word and $CounterMap$ is used to store the counts of French and English word pair. The results for these heuristic models are provided in the $Analysis$ section later.

IBM Model 1: As part of this model, Expectation Maximization algorithm is applied to compute the translation probabilities given a parallel corpus in two languages. Let a denote the alignment for a pair of sentences, and f and e denote the French sentence and English sentence respectively. Let J denote the number of words in French sentence. Then, translation probability is given by,

$$P(f,a|e) = \frac{\epsilon}{(J+1)^I} \prod_{j=1}^{J} \theta(f_j|e_{a_i}) \tag{1}$$

An important thing to note is that "null" is used for aligning a French word with no English word. The alignments with maximum θ values are the outputs of the algorithm. However, in this algorithm, all alignments are treated equally likely regardless of how far the indexes of words are in French and English sentence. This is often not a good idea because neighbouring words are more likely to be near each other and HMM based algorithm solves this.

Regular and Intersected IBM Models are implemented and the results for both are provided in the $Analysis$ section. $IBM1$ class implements the IBM Model 1 algorithm in forward direction and it uses $HashMap < String, Double >$ to store the translation probabilities for each pair of English and French word and $IntersectedIBMModel1$ class implements the intersected Model which ultimately retains the set of common alignments from both sides (forward and backward) to output the final alignment. An $OpenAddressHashMap$ implementation was also experimented with from the Assignment 1, but the results in terms of speed and memory were comparable.

HMM based Model: The HMM based alignment models capture the idea that neighbouring words in source language often align with neighbouring words in target language. Similar to IBM model, it calculates the translation probability θ but also accounts for the jumps in alignments with jump probabilities given by ψ which is also learned as part of the Expectation Maximization algorithm.

Both Regular and Intersected HMM Models are implemented and the results for both are provided in the $Analysis$ section. $HMMFrenchGivenEnglish$ and $HMMEnglishGivenFrench$ classes implement the forward and backward respectively and the $HMMAlignerIntersected$ class implements the intersected HMM Model which ultimately retains the set of common alignments from both sides to output the final alignment. Both the classes use $HashMap < String, Double >$ to store the translation probabilities for each pair of English and French word, and an array of $Double$ to store the ψ values. The probability which is to be maximized overall is a product of product of alignment probability and product of transition probability and is given by:

$$P(f, a|e) = \prod_j p(f_j|e_{a_j}) \prod_j p(a_j|a_{j-1}) \tag{2}$$

Apart from the basic implementation, α smoothing is also performed and analysed.

III. Experiments & Analysis

Heuristic Model: Following are the statistics for the Heuristic based Models and the least AER and highest BLEU scores for both 1,000 and 10,000 test sentence pairs were obtained using the Dice Coefficient approach.

Heuristic Method	Num Test Sente-nces	Train Time (in secs)	AER	Deco-ding Time (in secs)	BLEU	Mem-ory Used
MPI	1000	<1	0.7230	7.039	5.785	832M
MPI	10000	1	0.6003	20.163	12.110	838M
Dice Coeficient	1000	<1	**0.5240**	4.034	**8.617**	834M
Dice Coeficient	10000	1	**0.4637**	17.166	**15.885**	842M
Ochichai	1000	<1	0.5569	3.017	7.456	830M
Ochichai	10000	1	0.5304	9.675	13.684	841M

IBM Model 1: Following are the statistics for the IBM 1 based Models and the least AER and highest BLEU scores for both 10,00 and 10,000 test sentence pairs were obtained using the Intersected approach. It was also noted that converting all the words to **lower case** before indexing them improved both the AER and BLEU scores of all the models, and also resulted in less memory usage and thus, case conversion was performed in both IBM and HMM based models thereafter. Assigning ”null” a separate probability was also experimented with but the results were comparable to the Intersected approach. Memory used by the models for 10,000 sentences was about 850M. Moreover, the backward model performed better than the forward model for both IBM and HMM based model.

IBM Model 1	Test Sent-ences	Train Time (secs)	AER	Decoding Time (secs)	BLEU
IBM1 F->E	1000	3	0.4254	3.580	9.963
IBM1 F->E	10000	17	0.3415	13.910	15.005
IBM1 E->F	1000	3	0.4636	7.765	8.719
IBM1 E->F	10000	15	0.3875	**16.675**	16.305
IBM1 Intersected	1000	7	0.3420	5.265	8.185
IBM1 Intersected	10000	12.2	**0.2721**	15.141	**14.113**

HMM Model: Following are the statistics for the HMM based Models without α smoothing and, the least AER and highest BLEU scores for all 10,000 and 1,000 test sentence pairs were obtained using the intersection approach. ”null” was assigned a separate probability along with the Intersected approach and it resulted in significant improvement in AER. Memory used by the models for 10,000 sentences was about 900M.

HMM Models	Num Test Sent-ences	Train Time (in secs)	AER	Deco-ding Time (in secs)	BLEU
F ->E	1000	72	0.3576	4.580	12.210
F ->E	10000	678	0.3034	21.765	17.912
E ->F	1000	78	0.3489	7.765	11.259
E ->F	10000	621	0.3016	23.772	**18.480**
Intersected	1000	144	0.2504	5.413	11.506
Intersected	10000	1187	**0.1993**	18.987	**17.929**

Two different methods were experimented with to initialise the θ values for HMM based models. One is uniform probability and another by training IBM Model 1 for 1 iteration. The latter performed better than the former. The result comparisons are noted below on 10,000 test sentences:

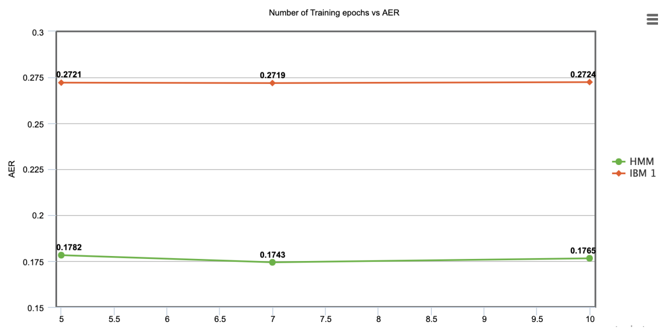
Intersected HMM Models	Train Time (in secs)	AER	Decoding Time (in secs)	BLEU
Uniform Init	1202	0.2175	19.986	17.085
IBM Init	1187	**0.1993**	18.987	**17.929**

Apart from this, various values of epsilon (ϵ) ranging from 0.005 to 0.2 were experimented with for assigning prior ”null” probabilities, and the best results were obtained for **0.01**.

Based on the paper (Och Ney, 2000), an approach of smoothing the transition table while training the HMM models was also tried and it improved the performance of the system in terms of AER although the BLEU score dropped down a little because of reduced overall recall. The comparison for the same is noted below on 10,000 test sentences:

Intersected HMM Models	Train Time (in secs)	AER	Decoding Time (in secs)	BLEU
Before smoothing	1187	0.1993	18.987	17.929
After smoothing	1843	**0.1744**	16.694	16.498

Overall, the AER results for HMM improved as the number of training iterations were increased until a point, and this can be seen in the following plot. **7** iterations gave the lowest AER score of **0.1743** However, for IBM Model, the AER was saturated in **5** iterations itself. But, the overall time to train also increases proportionally.



Additionally, it should also be noted that as the number of test sentences increased, the AER and BLEU scores also improved. The comparative results are shown in the Performance section later.

IV. Performance

The following are the best performance metrics reported for all the models trained on 10,000 test sentences on MacBook Pro 1.4GHz Intel Core i5 Processor with JDK 1.8:

Model Used	BLEU Score	AER	Decoding Time (in Secs)	Memory Used
Heuristic	15.885	0.4631	17.166	842M
IBM 1	14.113	0.2721	15.141	846M
HMM	16.694	0.1743	16.498	902MB

Following are the performance metrics on 100,000 test sentences:

Model Used	Trai-ning Time	BLEU Score	AER	Decoding Time (in Secs)	Memory Used
Heuristic	16 secs	19.769	0.4264	36.762	924M
IBM 1	753 secs	19.808	0.2215	111.170	846M
HMM	2 hours 45 mins	**22.490**	**0.1357**	61.164	1.3G

Overall, intersected HMM Model with alpha smoothing (α = 0.4) for which θ values were initialized with training IBM Model-1 for 1 iteration performed the best.

V. Conclusions

As part of the assignment, different models were discussed for fetching the best word alignments given a set parallel sentences in two different languages. The Intersected HMM model outperforms the other two models by incorporating the idea that neighbouring words in a source language sentence align with neighbouring words in target sentence. In future, a discriminative re-ranking approach can be applied over all these three models to assign weights to them appropriately and get the best alignment in a joint manner, however it couldn't be done due to lack of time.

References

[1] Vogel, Ney and Tillmann,"HMM-based Word Alignment in Statistical Translation" in COLINGVolume 2: The 16th International Conference on Computational Linguistics, 1996
[2] Och, Franz Josef and Ney, Hermann, "A Comparison of Alignment Models for Statistical Machine Translation," in COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics, 2000
[3] Svante Janson, Jan Vegelius "Measures of Ecological Association",1981
[4] Lee R Dice, "Measures of the amount of ecological association between species", 1945
[5] Church, Kenneth Ward and Hanks, Patrick, "Word Association Norms, Mutual Information, and Lexicography", in 27th Annual Meeting of the Association for Computational Linguistics, 1989
[6] Jurafsky, Dan. "Speech language processing." Pearson Education India, 2000.