NLP-267: Homework 1 - Word Alignment (Winter 2023)

Kushagra Seth
UC Santa Cruz
2005986
kuseth@ucsc.edu

Brian Farrell
UC Santa Cruz
2006370
bfarrel1@ucsc.edu

Mridul Pankaj Khanna
UC Santa Cruz
2006383
mrkhanna@ucsc.edu

Bhrigu Garg
UC Santa Cruz
2006380
bgarg@ucsc.edu

1 Motivation

Aligning words is a key task in machine translation. We start with a parallel corpus of aligned sentences. *For example*, we might have the following sentence pair from the proceedings of the bilingual Canadian parliament:

F: le droit de permis passe donc de 25 à 500.
E: we see the licence fee going up from \$25 to \$500.

The main *motivation* behind using IBM Model 1 over the Dice coefficient is that it provides a probabilistic estimate of the likelihood of a Target(English) sentence given a Source(French) sentence, which can be used to rank and select the most likely translation.
In contrast, the Dice coefficient simply measures the overlap between two sets of data and does not provide a probabilistic estimate of the likelihood of a translation. IBM Model 1 also takes into account not only the overlap between words but also the order of the words, and it uses a statistical approach based on counting co-occurrences of words in parallel texts to estimate the probabilities.

2 Description of Model

2.1 IBM Model 1

The main goal of the IBM Model 1 is to model the conditional probability $p(e|f) = p(e_1, \ldots, e_m|f_1, \ldots, f_l, m)$ where $e_1, \ldots, e_m$ is the Target(English) sentence and $f_1, \ldots, f_l$ is the Source(French) sentence [2]. The IBM Model 1 makes direct use of the idea of *alignments*, and the parameters of the IBM Model 1 are estimated using the Expectation Maximization (EM) algorithm.

It defines a conditional distribution:

$$p(e|f) = p(e_1, \ldots, e_m, a_1, \ldots, a_m|f_1, \ldots, f_l, m)$$

where $a_1, \ldots, a_m$ is the alignment of Source(French) sentence with words $f_1, \ldots, f_m$ [2]. IBM Model 1 uses only translation parameters $t(e|f)$, which are interpreted as the conditional probability of generating English word e from French word f.

2.2 EM-Algorithm

It is easy to calculate the estimations given fully-observed data. However, there are situations when we'll need to identify parameters because the data may be insufficient in various ways. When some data is missing, the EM algorithm is an effective iterative method to determine the Maximum Likelihood Estimate(MLE).

The EM algorithm is iterative and always improves a parameter's estimation through its process. It could be applied even when part of the data is incomplete. It is able to guess and estimate a set of parameters for your model under many situations.

However, the EM Algorithm has one *drawback*, which is that the parameters' initial values are chosen randomly. As a result, the algorithm may become trapped in a local maximum rather than the optimal global maximum. The EM algorithm can sometimes occasionally be quite slow.

2.2.1 Description of Implementation

Our implementation of IBM Model 1 includes two parts: training and testing. The training is done by running 10 iterations of the EM Algorithm. The testing part includes assigning alignment to each sentence pair with the highest $t(e|f)$ score, i.e., $a_i = \underset{j \in 0 \ldots l}{\operatorname{argmax}}\, t\left(e_i|f_j\right)$ [2].

2.3 Data Structures and Functions used

Our implementation of IBM Model 1 consists of the following data structures and functions [3]:

Class Alignment: A class that stores information about the word alignment between two sentences, the source sentence and the target sentence. The class has two attributes, f_sent and e_sent, which are lists of words in the source and target sentences, respectively. The class also has two dictionaries, f_index and e_index, which store the indices of each word in the source and target sentences.

Function em_ibm1: The EM (Expectation Maximization) algorithm implementation for IBM Model 1. It takes a list of sentence objects and the number of iterations as inputs. It returns a dictionary of translation probabilities $t(e|f)$, where e is a word in the target sentence, and f is a word in the source sentence. The function uses the dictionary count to store the intermediate results of the EM algorithm and the dictionary total to store the normalization values.

Function print_alignments: A function that prints the alignments between the source and target sentences based on the translation probabilities. It takes the list of sentence objects and the dictionary of translation probabilities as inputs and returns None.

Function main: The main function reads the data, trains the IBM Model 1 using the EM algorithm, and prints the final alignments. The function reads in the source and target sentences, create Alignment objects for each sentence pair, and trains the IBM Model 1 by calling the em_ibm1 function. Finally, it prints the alignments by calling the $print\_alignments$ function.

3 Results

Table 1: Performances of IBM Model 1

Iterations	Precision	Recall	AER
1	0.213	0.260	0.771
2	0.442	0.65	0.48
3	0.503	0.724	0.425
4	0.538	0.757	0.391
5	0.546	0.766	0.383
6	0.558	0.781	0.370
7	0.563	0.781	0.367
8	0.564	0.784	0.365
9	0.567	0.786	0.362
10	**0.572**	**0.789**	**0.357**

Table 2: Dice Coefficient Model vs IBM Model 1

	Precision	Recall	AER
Dice Model	0.243	0.54	0.68
IBM Model 1	0.572	0.789	0.357

The AER for IBM Model 1 gradually decreases as the number of iterations for the EM Algorithm increase. This is reasonable since the EM algorithm always improves a parameter's estimation through its process.

4 Code and Repository

The code we used to train and test our IBM Model 1 against Dice Coefficient Model is available at: https://github.com/bhragu/Machine-Translation-NLP-267/tree/main/hw-1.

References

[1] Jhu-2014 writing exercise documentation.

[2] Michael Collins. Statistical machine translation ibm models 1 and 2.

[3] Chris Callison-Burch Philipp Koehn. *LaTeX: Statistical Machine Translation*. 2008.