Bilingual Continuous Space Language Model

Hieu H. Pham – hyhieu@cs.stanford.edu
Hung Tan Tran – htran1@stanford.edu

Abstract

We attempt using simple feedforward neural networks to learn a continuous space language model (CSLM) and evaluate its performance on French and English. The training of the model is performed bilingually, in which we try to capture not only different features from one language, but also the multilingual relation demonstrated in the parallel training data. We achieved the perplexity of 191.2 on French and 193.5 on English, and visualized some interesting monolingual and bilingual relations.

1 Introduction

Language Model (LM) is a tool that scores the fluency of sentences in a specific language. LMs arise in many different problems of natural language processing, such as speech recognition, handwriting recognition, or machine translation (Goodman J., 2001). A popular approach for LMs is N-gram, in which the model makes the assumption that a word w_i in a sentence depends only on its $N-1$ precedents. That is

$$p(w_1 w_2 ... w_n) = \prod_i p(w_i | w_{i-1} ... w_{i-N+1})$$

N-gram LMs estimates from a training data set the probabilities $p(w_i|w_{i-1}...w_{i-N+1})$, and then based on these learned probabilities to score the new sentences. This method suffers many shortcomings. First, it does not generalize well: if we have a combination of words that has not yet appeared in the training data set (which very likely happens), the sentence will be scored 0. Many smoothing techniques have been developed to overcome this issue, such as Kneser-Ney smoothing (Ney et al., 1994). However, a bigger issue is that for an N-gram model, we would have to store $|V|^N$, where V is the vocabulary of a language, whose reasonable size is about $100K$. With this requirement, even tri-gram model is a memory challenge.

An alternative to this approach is that instead of explicitly storing all $p(w_i|w_{i-1}...w_{i-N+1})$, we store parameters of another model, based on which we can compute these probabilities. A popular idea is proposed in Bengio, Y. et al., 2003, called the Continuous Space Language Model (CSLM). Compared to N-gram models, CSLMs are not only more accurate on unseen context in training corpus but also more memory efficient. The CSLMs of Bengio, Y. et al., 2003 uses a neural network to learn a map from the vocabulary of the language into a high dimensional vector space, and then based on this representation of words to compute the N-gram probabilities.

Since Bengio, Y. et al., 2003 proposed CSLM, the model has been very popular and successful, such as Le H.S. et al, 2010, or Mikolov, T. et al., 2013. We are particular motivated by the work of Mikolov et al., 2013, in which they built a recurrent neural network (RNN) to exploit similarities among different languages, even ones with dissimilar structures. Although their work shows very promising results of RNN LMs, the model is slow to train. In this paper, we attempt a different approach to a subset of their tasks. Specifically, we focus on the task of French-English translation. Using a very simplified, and hence efficient, structure of feed forward neural network (FNN), we achieved similar results as in Mikolov et al., 2013.

2 Bilingualized Neural Network

Our model is based on the CSLM described in Schwenk H. et al., 2010. We use an FNN with

two layers, which we will call the monolingual FNN. Supposed that we have a language with N words in the vocabulary. We will learn a matrix $\mathbf{W} \in \mathbb{R}^{N \times P}$, whose row i^{th} corresponds to the representation of word i^{th} in the vocabulary. When we have an n-gram $w_1 w_2 ... w_n$, at the first layer, called the *shared projection layer* of the NN, we pick out the corresponding word vectors W_{w_i} and concatenate the first $n-1$ of them in to a $(n-1)P$-dimensional vector, which is given to the neural network as the input. Let this vector be $c \in \mathbb{R}^{(n-1)P}$.

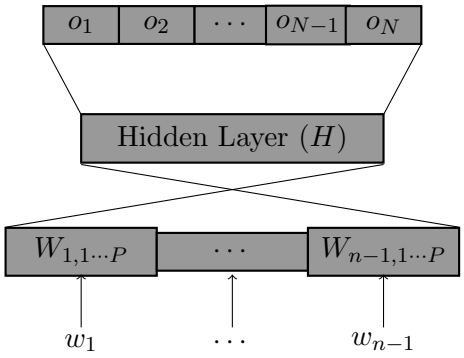

Figure 1: Structure of a monolingual FNN

At the second layer, called the *hidden layer*, we used hyperbolic tangent function to compute $d = \tanh(\mathbf{M}c + b)$, where $\mathbf{M} \in \mathbb{R}^{(n-1)P \times H}$, with H is the size of our the hidden layer, and tanh is applied component-wise to the vector. The output layer of the NN has size N, equals to the size of the language's vocabulary. At this layer, we first compute $o = \mathbf{V}d + k$, and then we add a softmax layer p, where $p_j = \dfrac{\exp(o_j)}{\sum_{j'=1}^{N} \exp(o_{j'})}$ represents the conditional distribution

$$p_j = p(w_n = j | w_1 w_2 ... w_{n-1})$$

The objective that the NN tries to minimize is

$$E(\theta) = \sum_{i=1}^{N} y_i \log p_i + \frac{\lambda}{2} \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2 \right)$$

where y is the output vector, for which $y_{w_n} = 1$ and all other components are 0. Thus, the first summation in this objective function is the cross-entropy of the model on it training data. Only one term in this summation is non-zero, and this eases the work of evaluating gradient at the NN's output layer. The second term, with weight decay factor λ is there to prevent overfitting issues. Training is done by stochastic gradient descent. That is, for each training example, we do a standard feed forward and then back propagation to compute the corresponding gradients $\nabla_\theta E$, and finally update θ via the equation

$$\theta := \theta - \alpha \nabla_\theta E$$

where α is the learning rate. The parameters α and λ are determined at the construction of our NN, and hence could only be tuned after a few experiments. In our final experiments, we set $\alpha = 5 \cdot 10^{-4}$ and $\lambda = 1.2$.

2.1 Bilingualization

Adapting the bunch mode idea from Schwenk, H., 2010, we propose a bilingual setup for the training phase of our NN. Instead of having one matrix W as described above, we used French-English parallel training data. We doubled the size of our matrix W, to simultaneously hold the vocabularies of both languages. Doing so drastically increases the size of our NN, in the sense that the sizes of the matrices $\mathbf{M}$ and $\mathbf{V}$ exploded by $2H$. However, as a tradeoff, our NN has more synapses, and thus, is capable of capturing similar structures between the two languages via parallel linguistic data.

Bilingual training also leads to a change in our objective function: we have to capture the pair of next words in our n-gram, namely the next English word and the next French word. Our new objective function is therefore

$$\tilde{E}(\theta) = \sum_{i=1}^{N} y_i \log p_i + \sum_{i=1}^{N} \tilde{y}_i \log \tilde{p}_i + \frac{\lambda}{2} \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2 \right)$$

where the terms $\tilde{y} \log \tilde{p}_i$ correspond to the cross entropy of the model for French. As for the case of $y_i \log p_i$, only one $\tilde{y}_i$ is one, and all the others are zeros.

Last but not least, in order to keep the size of the vocabulary (and hence the size of our NN)

reasonable and manageable, we restrict the size of vocabulary of both languages into $N = 10K$. The word in the vocabularies of French and English that we chose are the words which appear most in the training data set. For each language, we also reserved a special token, called `<UNK>`, which generally represents the words that are not in the vocabularies we are considering. Linguistically, this corresponds to the action that we treat the words we are less likely to see the same. In our experiments, the full size of the English and French vocabulary is $1.1M$, and a simple count through out the training dataset yields that 90% of them appears less than 10 times in the data.

2.2 Implementation notes

Our implementation of the NN is array-based. In our code, we explicitly used the arrays of doubles W, c, b, M, V and k. The sizes of these arrays are specified at the construction of the NN, making the NN scalable. However, we restricted the NN into a three layer NN as described above, so it is not adaptable for generic uses of NN. As a tradeoff, we achieved faster training time, for the avoidance of creating many classes for different NN routines. Also, we use a 5-gram model, in which each word is mapped into the space with dimension $P = 50$, and we set $H = 100$ neurons in the hidden layer.

3 Experiments

3.1 Perplexities

We evaluate our model by measuring its perplexity on an unseen data set `dev`. Let $a_{1,1...5}$, $a_{2,1...5}$, ..., $a_{n,1...5}$ be the 5-grams in our data set, then the perplexity of our model on this data set is computed via the geometric mean

$$PPL = \left(\prod_{i=1}^{n} p(a_{i,5} | a_{i,1} a_{i,2} a_{i,3} a_{i,4} \right)^{1/n}$$

Due to the structure of our NN, we can only compute parallel testing data. That is, when evaluating perplexity of the model, require two parallel pairs of 5-gram, from French and English, then compute p and $\tilde{p}$ at the same time. This allows us to simultaneously compute the perplexity of French and English sentences in the test corpus. We then use the geometric mean of them as the final perplexity of our model.

3.2 Experiment settings

We extracted our training and testing data from the Europarl French-English bilingual corpus provided by Koehn, P., 2005. The training data set consists of $100K$ parallel sentences, and the testing corpus consists of 100 parallel sentences. To extract our training data, we randomly picked $100K$ parallel sentences from the corpus, then preprocess them into lowercase words and separate punctuations from words (such as "`Parlement, debout,`" into "`parlement , debout ,`"). We then train our NN on the extracted data set for 7 epochs. After each epochs, we compute the perplexity of the model. If the perplexity does not decrease after an epoch, we stop the training process. Otherwise, we save our model into a file, and continue with the next epoch.

All training and testing are performed on our personal laptop computer, MacBook Pro running Mac OS X Maverick, with the Processor 2.3 GHz Intel Core i7 and 8.0 GB of RAM. In our experiments, each epoch runs in about 4.5 hours, and all 7 epochs are carried out.

4 Results

4.1 Perplexities

We observed decrease in perplexities after each epochs. The result is shown in the table below

Epochs	Perplexities		
	French	English	Total
1	$8.16E11$	$1.16E12$	$9.73E11$
2	35099.3	33011.2	34039.2
3	7245.6	8312.8	7760.8
4	2932.4	3123.5	3026.4
5	701.3	604.2	650.9
6	241.2	234.1	237.6
7	191.2	193.5	192.3

Table 1: Perplexities of the model after each epoch

4.2 Visualization

Further than achieving the perplexities above, our model also captured many linguistic interpretations of both French and English. Following the idea of Mikolov, T., et al., 2013, we implemented the Principle Component Analysis (PCA) to project the word vectors from the 50-dimension space into the plane. This shows not only interesting relations in French and English themselves, but also between the two languages. Following, we plot some interesting relations that we have found. The plot was carried out by first doing PCA on specific words to produce their corresponding 2-dimensional vectors, and then using `Geogebra` to plot those points onto a plane.

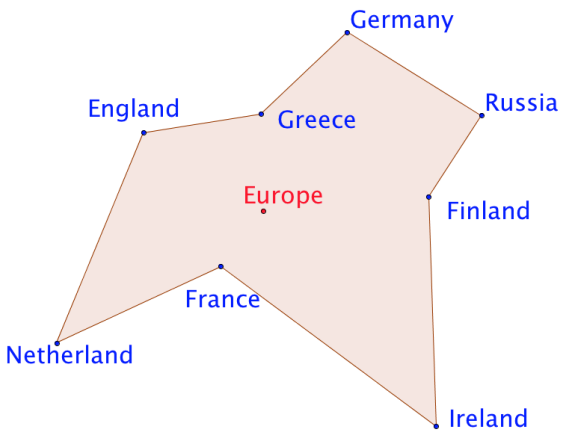

Figure 2: The vector `Europe` appears to be closed to the centroid of the vectors `England`, `France`, `Germany`, `Ireland`, `Greece`, ...

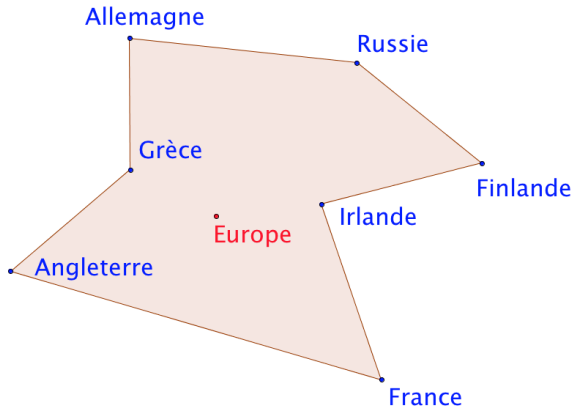

Figure 3: The same set of countries in French produces roughly the same figure with that in English, still with `Europe` closed to their centroid.

Perhaps one of the most amazing achievements of our model is that similar set of countries in French produces roughly the same figure. Actually, two figure differs by roughly a rotation, and most significantly, the polygon formed by those countries have the same counter-clockwise order of vertices.

Since we projected all words from the two vocabularies into the same linguistic space, there are also many relevance between two those vectors. For example, in the 50-dimensional space, we found out that the vectors `one-un`, `two-deux` and `three-trois` are roughly parallel (their component-wise difference roughly align into the same ratio).

5 Conclusion

Compared to the perplexities of less 100 of state-of-the-art language models, ours are considered modest. Further than that, we required bilingual training and testing data, which are generally harder to get than monolingual data. More problematic is the decoding phase of the neural network: our NN can only evaluate the n-gram probabilities if it is given parallel input. Finally, the worst issue is that although our model is intentional programmed based on array to speed up the training time, its training phase turned out to be prohibitively slow. One way to speed up the training phase is to use the bunch-mode optimization of mentioned in Schwenk, H., 2010, but that would require a lot of modifications in our code to allow cache-friendly matrix multiplication, as well as other caching techniques. Thus, there are still a lot of possible improvements needed for our model.

However, we have shown that the model proposed by Schwenk, H., 2010, can be adapted to train on bilingual data and could indeed capture interesting bilingual features. Perhaps after adding powerful optimization techniques that allow more efficient training, we can train our model on k-lingual data, and promisingly, can achieve better results.

6 Acknowledgement

In this last section, we want to deliver our grateful words to Professor Christopher Manning and Professor Andrew Ng, who have given us two interesting and helpful courses, namely cs224n and cs229. Without them, this work would have no reason to be done, and hence could never be done.

We also thank the TAs of the two classes, who have sent out a lot of emails to remind us to stay on track. Among these awesome graduate students, we thank Minh-Thang Luong for giving us many helpful advice and references.

References

[1] Joshua Goodman, *A Bit of Progress in Language Modeling*. Microsoft Research One Microsoft Way Redmond, WA 98052, 2001

[2] Holger Schwenk, *Continuous space language models*. Spoken Language Processing Group, LIMSI-CNRS, BP 133, 91403 Orsay cedex, France, 2006

[3] Philipp Koehn, *Europarl: A Parallel Corpus for Statistical Machine Translation*. MT Summit, 2005

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, *A Neural Probabilistic Language Model*. Journal of Machine Learning Research 3 pages 1137–1155, 2003

[5] Tomas Mikolov, Quoc V. Le, Ilya Sutskever, *Exploiting Similarities among Languages for Machine Translation*. arXiv, 2013