NLP ENSAE ParisTech - Language Modeling - Spring 2019

Authors: Dorian Baudry & Alexandre Filiot.

1. Introduction

For this project, we chose to study language modeling. LM goal is to assign probability to a text. It is used in various applications: speech recognition, machine translation, OCR... A current metric to evaluate the qualities of the latter tasks is the perplexity (hereinafter PP). As suggested by (Jurafsky & Martin, 2014), "an intrinsic improvement in perplexity does not guarantee an extrinsic improvement in the performance of a language processing task. Nonetheless, because perplexity often correlates with such improvements, it is commonly used as a quick check on an algorithm". Here, **perplexity is our main metric of interest and quantifies the ability of our models to predict the next word**, a group of words or a sentence, based on previous information.

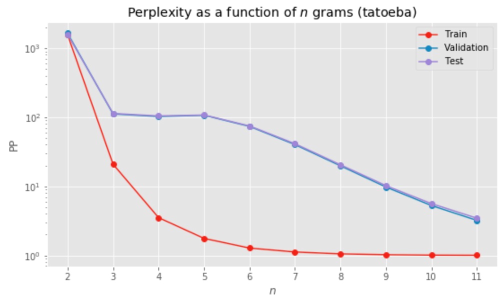
2. Guidelines

Our project focuses on **word level** LM models. As done in class, we wanted to look over the different models encountered in language modeling, from the most basic ones to deeper models. To this extent, we have implemented • ***n*-gram models and their variations (smoothing, backoff, interpolation)** from scratch • **recurrent deep networks (RNN, LSTM, GRU, BiLSTM) and feed foward ones (neural *n*-grams) with Keras**. Thus, our project aims at putting theory into practise. To do so, we consider the 2 word level datasets available (`wiki` and `tatoeba`). The models are compared according to their perplexities on validation (tuning) and test full sets. Finally, **we have also tested those models on text generation task**.

3. n-grams

As a very beginning, we implemented classical n-gram models. From our observations, n-grams suffer from 3 main caveats. The first one is its generalization capacity. Some longer testing n-grams (e.g. $n > 4$) don't appear in training, which makes PP tends to ∞ as the corresponding probabilities are zero. To tackle this issue, we used the Laplace expression with both counts equal to 0, namely, $-1/V$ with V the vocabulary size. This approximation allowed us to represent the perplexity as a function of n (see figure 1). The second issue comes from underflow. Despite the log-sum-exp trick, some training sentences have 0 probability (see notebook for examples) because of their length. Last but not least, **the perplexity (on train or test) tends to** 1 **as probabilities tends to** 1 **with increasing** n. Indeed, counts become rather null or equal to 1, which makes probabilities all equal to a constant $C = \#\{n-1-\text{contexts}\}/\#\{n-\text{grams}\}$ which tends to 1 with n.

Figure 1. Perplexity as a function of n-grams (`tatoeba` dataset)

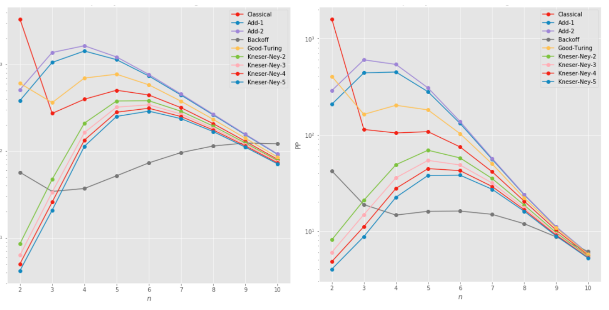


n-grams are surprisingly good at generating text. To generate next words from, say, 6 starting words, we recursively used 5-grams models, then 4-grams if no context could be found...until 2-grams. If no bigrams exist, then we draw the next word randomly among unigrams. Here are some examples of generated words (bold) based on `tatoeba` dataset: `I am happy` **`to meet you`** – `I need you` **`to do that`** – `what are you planning` **`to stay </s>`** – `the cat is` **`playing with the ball </s>`** – `The dogs eat` **`the last orange`**. Those propositions are more meaningful than those obtained with a training on `wiki` dataset as the latter has been obtained using crawling. `Tatoeba` is a collection of basic sentences that best fit to our expectations (basic predictions).

4. Variations of n-grams

We also implemented Laplace and Kneser-Ney smoothings, stupid backoff and Good-Turing estimation. Figure 2 shows the evolution of perplexity on test and val sets as a function of n (`tatoeba`). Values and graphs similar to figures 1 & 2 can be found in the notebook and `README.md` file. It is first interesting to notice the convergence to 1 as a function of n-grams. We are more sceptical about our implementation of Kneser-Ney smooth-

Figure 2. Perplexity as a function of n-grams (`tatoeba` dataset). Left: validation, right: test set.



ing as perplexities are extremely low for short n-grams. Despite this, the text generation is quite the same with some improvements, as for Kneser-Ney smoothing which makes new meaningful propositions. To take the previous sentences, we now have now with the latter (other sentences are displayed in the notebook): `I am happy` **`to meet you`** – `I need you` **`to do that for us`** – `what are you planning` **`to do </s>`** – `the cat is` **`playing with the ball </s>`** – `The dogs eat` **`the last orange`**.

5. Deep networks

In order to compare with the latter models, we implemented with Keras some of the architectures studied in class: Recurrent (RNN), Gated Recurrent Unit (GRU) and Feed Forward (FFNN) networks; LSTM, BiDirectional LSTM (BiLSTM) and a fancier architecture named BiLSTM2DCNN[1]. Model selection was based on validation PP for the following parameters: • `ngram` $\in \{2,3,4,5,6\}$ (length of n-grams) • `mcells` $\in \{16,32,64,128,256,512,1024,2048\}$ (number of memory cells in recurrent layers) • `dropout_rate2` $\in \{0,0.1,0.2,0.3,0.4\}$ (dropout rate after recurrent layer) • `embedding` $\in \{10,20,50,100,200,300\}$ (size of embedding space). Concerning the latter, we also tried Fast-Text pre-trained 300-sized embedding from (Joulin et al., 2016). Figure 3 well reflects the influence of those parameters on the whole bunch of networks. Namely, perplexities and accuracies are optimal for $n = 4$; improve with embedding size and number of memory cells. Dropout reduces overfitting, in particular for larger networks (see notebook: *4.3. Influence of parameters*), but does not improve performance. Models were selected on `tatoeba` data set with a batch size of 256 in order to save computational resources. We didn't fine-tune on `wiki` data set but re-trained the best architectures on it later on. The perplexities are reported in table 1. They were computed as $\exp \bar{H}(W)$ with $\bar{H}(W)$ the cross-entropy averaged on the batch-size we set to 1. Regarding text generation, results are still more satisfying on `tatoeba` (which is logical as we fine-tuned on it) but less than before, with a tendency to loop over stop words. Now we have the following: `I am happy` **`to be a <unk> to`** – `I need you` **`to do that by themselves`** – `what are you planning` **`do that by themselves did`** – `the cat is` **`very good at doing that`** – `The dogs eat` **`<unk> of the <unk> of`**.

[1]See Peng et al. (2016), "Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling".

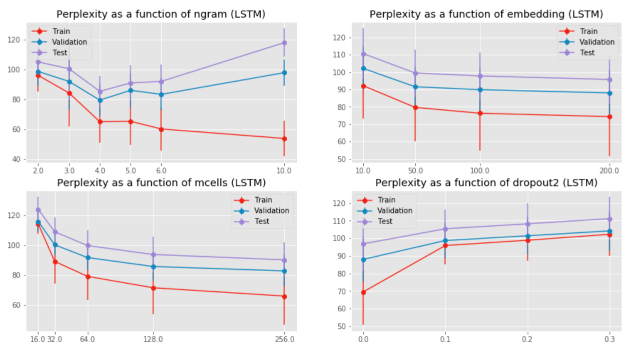


Figure 3. Influence of parameters on perplexity and accuracies (LSTM, `tatoeba`).

Table 1. PPs on test and val full sets for deep models.

	TATOEBA		WIKI	
	VAL	TEST	VAL	TEST
LSTM (512)	62.0	67.4	155.4	157.6
BILSTM (2048)	**59.2**	**63.7**	**144.3**	**145.2**
RNN (512)	66.5	71.1	172.1	174.5
GRU (256)	61.2	66.7	161.5	162.4
FFNN (2048)	103.4	117.0	512.3	594.0
BILSTMCNN (512)	66.0	70.8	167.1	166.4

6. Feedback & Discussion

Perplexity on his own does not fully characterise the quality of our models. That's why, before concluding, a model's improvements in PP need to be confirmed with some on a real LM task. Poor results on n-grams (which may be due to our implementation) contrast with their good ability to predict text, and conversely for deep networks. The use of recurrent layers, especially BiLSTMs (BERT's core), offer dramatic performance gains compared to feed-forward networks. More realistic data set as `wiki` was harder to exploit. As an opening, one could have tested a batch size of 1; transformers and focus more on Kneser-Ney smoothing.

References

Joulin, A., Grave, E., et al. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

Jurafsky, D. and Martin, J. H. Speech and language processing - chapter 4: N-grams. 2014.