LSTM for Text Generation and Classsification

Frédéric Boileau FREDERIC.BOILEAU@UMONTREAL.CA
Jimmy Leroux JIMMY.LEROUX@UMONTREAL.CA
Nicolas Laliberté NICOLAS.LALIBERTE@UMONTREAL.CA

Abstract

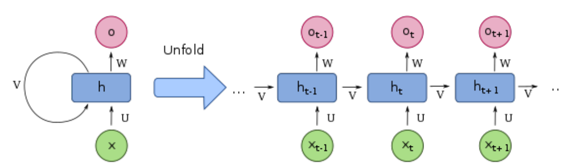
In the context of our project for the IFT6269-A2018 we have investigated the different available models for the generation and classification of natural text data. While we were initially interested in fully probabilistic models such as Hidden Markov Models (HMM), a quick review of the contemporary literature on the topic of pattern recognition on sequences made it clear that neural networks (NN) provided the better toolset. In the end we implemented two Long Short-Term Memory networks, for generation and classification respectively which we trained on subsequences of some corpora of prose fiction. Despite the rather coarse nature of our implementation the generators were able to produce legible and decently structured text which reflected the material used for training; in the style of the writing for example. The classifier reported excellent metrics however there are several interesting questions as to what overfitting consists of when classifying prose fiction. The most obvious is the one of proper nouns, but others such as temporal markers are more subtle. The solutions to be decided with some context of the intent of the task, so as to be the answer to a well-defined question.

1. Introduction

Hidden Markov Models used to be the go-to probabilistic tool to reason about sequential data; the Markov assumption however proves to often be unreasonably strong. Adding links to form higher order chains is not a scalable solution as the computationnal complexity grows exponentially in the order of the chain. Recurrent Neural Networks (RNN) constitute a family of neural network architectures specialized to process sequential data which can forfeit Markovian assumptions while remaining tractable. RNNs can track longer range dependencies while staying tractable by leveraging the simple idea of sharing parameters across the model (Goodfellow et al., 2016). Concretely this means adding loops to the hidden units of the neural network. RNNs have been successfully used in diverse domains for generating sequences such as music and text (Graves, 2013). Despite the aforementionned features, naive RNNs suffer from the fact that the influence of some input to the hidden layer either decays or blows up exponentially in time, a phenomenon reffered to in the literature as the *vanishing gradient problem*. The cost of abandonning probabilistic models such as the HMM in favor of neural networks is the loss of a fully probabilistic interpretation. There has recently been an increased interest into finding reasonable probabilistic interpretaions to RNNs, see for example (Choe et al., 2017). On the other hand the very existence of some monolithic notion of "interpretability" has been recently questionned, see (Lipton, 2016) for a philosophically inclined take on the question.

2. Neural Networks Archicture

2.1. RNN and the Vanishing Gradient



$$h_t = \sigma(Ux_t + Vh_{(t-1)} + b_h) \quad o_t = \text{softmax}(Wh_t + b_o)$$

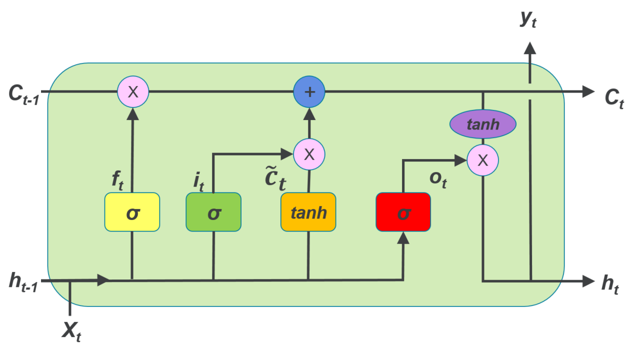
Where U, V and W are weights matrix and the vectors b are bias parameters. Consider the gradient of $o_{t+\delta}$ with respect to h_t. Applying the chain rule according to the graph above we get:

$$\nabla_{h_t} o_{t+\delta} = \left(\prod_{k=t+1}^{t+\delta} V^T \text{diag}(h_k(1-h_k)) \right) \nabla_{h_{t+\delta}} o_{t+\delta}.$$

Thus, as δ grows, the gradient grows exponentially with V. If V is small or large then the gradient will either vanish or explode. A myriad of solutions exist such as regularization through weight noise, the Long Short Term Memory architecture tries to tackle this issue on a higher level than regularization.

2.2. LSTM Architecture

To go from a RNN to a LSTM we replace hidden units with components known as *memory cells*. Our presentation and notation follows (Lipton, 2015).



Intuitively, RNNs have *long-term memory* in the form of matrix weights, they change during the training; encoding through training some general knowledge about the data. They also have *short-term memory* in the form of activation passing from each node to successive ones. The memory cell introduced in the LSTM model formalizes those notions and provides a framework to control their impact on the internal representation of the network.

- f_t, i_t, o_t: Respectively forget, input and output gates.
  - Sigmoidal units activated through x_t (current input) and h_{t-1} (past hidden unit output)
  - f_t controls the recursive flow
  - i_t and o_t control the input and output flow respectively
  - $h_t = o_t \odot \tanh(c_t)$ where $\odot$ denotes element wise multiplication.
- $c_t = \mathbf{f_t} \odot \mathbf{c_{t-1}} + i_t \odot \tilde{c}_t$: The cell which has a self-connected edge with a fixed unit weight, thereby delegating control of recursion to the gate

The architecture of the memory cell improves the capacity to learn long time dependencies. Indeed, following the definition of the cell state, we have

$$c_t = \sum_{k=o}^{t} f_t \odot \cdots \odot f_{k+1} \odot i_k \odot \tilde{c}_k.$$

We can see that the gradient may pass over long time gaps as long as the forget gates are open (closed to 1) which can be achieved with a appropriate initialization setting a large bias for the gates f.

3. Implementation

Two text sequences (referred to as datasets later on) were used for training: two concatenated novels of Jane Austen and of George Eliot respectively. All in English and utf-8 encoded. Punctuation and structure was left unprocessed. Each dataset was split in sequences of 50 tokens (i.e. "words") and a dictionnary was built from the complete input ($\approx 60k$ unique tokens), defining the input space for the networks. Vector encoding of this space was used through an embedding layer mapping the words to a real vector space of dimension 256. Available embeddings such as *word2vec* and *glove* were initally used but proved to be more cumbersome than our own trained version.

We trained four LSTM networks with the aforementionned data, a generator and a classifier, once without preprocessing and another where the NLTK Part of Speech Tagger (POS tagger) was used to filter out proper names and replace them with generic ones from the *names* dataset of the NLTK library. The generic name replacement was shared across the Jane Austen and George Eliott datasets. This was an effort to make the classifier be trained on rather structural aspects of the prose, such as syntax, length of sentences and broad notions of *style* as opposed to simply directly matching vocabulary which would have been trivial with the names of the characters.

Training was achieved at "word" level (tokenized with NLTK). Character level had been prevously envisionned for the rest of the project as it is more flexible and can *learn new words and structural information*(Graves, 2013) for the generators). Despite this we kept the training at word level for multiple reasons, the primary one being robustness towards unicode characters which might vary between versions of the text available and the second one being speed of convergence.

To generate the sequences the trained generators were initialized with a random word drawn from the dictionary and the most likely next word was fed back in the network until the desired sequence length was reached. 1000 sequences (250 per model) were generated.

The classifier were trained on the original datasets and were was then used to classify the generated synthetic data estimate which dataset (or model, equivalently) was used for its generation, thereby giving us some "metric" of the quality of the data.

4. Empirical Results

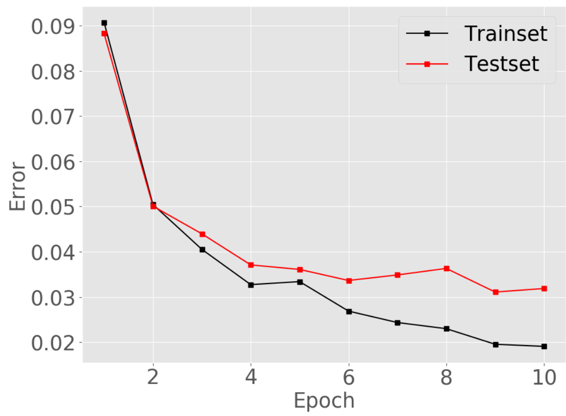


Figure 1. Classification error on train/test set.

Training trough minimization of the cross-entropy loss, which is equivalent to *perplexity*; the standard metric for language modeling (Graves, 2013).

Dataset	BPC	Perplexity
Harry Potter	1.00	33
LOTR	1.02	35
Random quotes	1.10	45
Shakespeare	0.94	26

Examples of generated quotes:

- “ well , we ’ ll do it with a wand , ” said hermione . “ really ? ” said harry , looking at each other .
- what looked about this way , the black citadel , was far from the darkness , the ring was heard , but the sea big was big , and a great ring was in his battle .
- failure is a beginning of love and a family which comes from god .
- ” that now my mind shall screens his music , ” ” and i to give thee my vulgar heaven , ” ” i am your brother .

5. Discussion

5.1. Numerical Concerns

5.2. Stylometrics

Authorship analysis is a field of computationnal linguistics where the author's identity is to be extracted from the writing style of text. (Ding et al., 2016). It has diverse applications beyond complementing the work of traditionnal philologists such a fraud and plagiarism detection or even identifying cybercriminals on underground forums(Afroz et al., 2014). Classically the field depended on statistical analysis of manually engineered features such as word frequency, length of sentences etc.

Despite its great importance and the fact that it is fundamentally quite well defined, i.e. the question who wrote this text is more well posed than the ones concerning sentiment analysis for example, there are major challenges. Clearly one needs to differentiate between writing features specific to writing style as opposed to context/ content specific features as Rosenthal and Yoon mention in their paper on the application of stylometric techniques to detect multiple authorship in legal decision (2011).

In this context the simple "anonymization" of the text sequences we performed has the intent to disentangle the contextual features from the imprint of the author. We have mentionned before that the classifier could be used as some kind of metric on the synthetic data. Conversely, suppose one is fairly confident in the quality of the generated text, i.e. after reading its content and using some domain knowledge experts determines that it reflects quite well the imprint of an author's style. In this case the generated data could be a candidate for training of classifiers to detect authorship, when the data is sparse for instance or simply because the very nature of synthetic data could prevent overfitting to less relevant features, the author's footprint being in some sense crystallized in the generator's features. Of course this is purely circular if both directions are considered at the same time and we have found the latter(using synthetic data to evaluate the generalization of the classifier) to be more interesting both in the results it yields and in its theoretical formulation.

In essence the goal would be to use text generation to trim from data all contextual information that is not considered to be relevant in determining the author (such as names or temporal and spatial markers in some cases) while keeping the semantics of the text intact. Afterwards using that stripped text to train the generators to obtain more synthetic data which reflects the author in its structural semantics, beyond crude word frequency. Of course the question as to what to keep from the original text, what to trim and generally what features constitute an author's footprint is one that for the moment relies heavily on domain-knowledge, however, given that knowledge, LSTM can leverage it to perform much deeper analyses than classical statistics provide, relying on structure and not just isolated features.

6. Conclusion and Further Work

conclu bla bla

7. Citations and References

References

Afroz, S., Islam, A. C., Stolerman, A., Greenstadt, R., and McCoy, D. Doppelgänger finder: Taking stylometry to the underground. In *2014 IEEE Symposium on Security and Privacy (SP)*, volume 00, pp. 212–226, May 2014. doi: 10.1109/SP.2014.21. URL `doi.ieeecomputersociety.org/10.1109/SP.2014.21`.

Choe, Yo Joong, Shin, J. C., and Spencer, Neil. Probabilistic interpretations of recurrent neural networks, 2017.

Ding, Steven H. H., Fung, Benjamin C. M., Iqbal, Farkhund, and Cheung, William K. Learning stylometric representations for authorship analysis. *CoRR*, abs/1606.01219, 2016. URL `http://arxiv.org/abs/1606.01219`.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618, 9780262035613.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`.

Lipton, Zachary Chase. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015. URL `http://arxiv.org/abs/1506.00019`.

Lipton, Zachary Chase. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016. URL `http://arxiv.org/abs/1606.03490`.

Rosenthal, Jeffrey S and Yoon, Albert H. Detecting multiple authorship of united states supreme court legal decisions using function words. *The Annals of Applied Statistics*, pp. 283–308, 2011.