Project Proposal For Statistical NLP 2019

Jiyuan Lu
Courant Institute of Mathematical Sciences
New York University
New York, NY 10025
jl111046@nyu.edu

1 Problem Setting

Natural Language Generation (NLG), also referred to as text generation, is a subfield of natrual language processing, which leverages knowledge in computational linguistics and artificial intelligence to automatically generate natural language texts. Text categorization, also referred to as text classification, is the task of assigning predefined tags or categories to text according to its content.

As a paricular application of NLG, automatically generating text that mimic what a real person says is an interesting yet challenging problem. Thanks to various modeling techniques scientists have developed during the recent years as well as abundant data available on the Internet, letting computers generate almost-real sentences that can easily fool people is no longer a wild dream.

President Donald Trump frequently uses social media, especially Twitter. From his offcial declaration of candidacy in June 2015 through the first two-and-a-half years of his presidency, he tweeted over 17000 times. By this time, the end of 2019, he should have posted many more tweets. Apart from the large quantity of his tweets, Trump's tweets has a very unique writing style. For example, he used an increasingly disengaged style over the course of the campaign - essentially presenting his opinion as facts and ignoring alternative viewpoints. These factors make Trump's tweets an ideal data source for training a text generator.

In this project, I am going to implement a style-mimicking text generator called Trump-Robot, which automatically generates tweets in the style of President Donald Trump. In the mean time, I will separately train a text classfier that properly distinguishes Trump's tweets and other people's tweets. Then I will use the text classifier to evaluate the performance of the text generator by seeing how the classifier can distinguish text made up by the text generator from real text. Furthermore, as a potential future work, I might consider using Generative Adversarial Network (GAN) to boost the performance of both the text generator and the text classfier.

2 Dataset

The main dataset is Trump's tweets, which is easily accessible thanks to various websites maintaining Trump's historic tweets. In particular, I will be using the dataset from a website called Trump Twitter Archieve. This dataset contains 27948 tweets of Trump from 5/4/2009 to 11/4/2019, and is used for training both the text generator and the text classfier.

The second dataset is random recent tweets selected from Twitter and can be acquired using the Twitter API. I will collect the same amount of random tweets as in the Trump tweets dataset, and it is used to train the text classifier only.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

3 Evaluation Metric

There are two models to evaluate in this project: a text generator for generating made-up tweets in the style of Trump, and a text classifier for distinguishing Trump's tweets from other people's tweets.

The text generator is essentially a language model. The best way to evaluate the performance of a language model is to embed it in an application and measure how much the application improves. Such end-to-end evaluation is called extrinsic evaluation. Extrinsic evaluation is the only way to know if a particular improvement in a component is really going to help the task at hand. Thus, for text generation, we can compare the performance of two generators by running a separately trained text classfier to distinguish made-up tweets from real Trump's tweets. The one which better fools the text classifier, i.e., whose made-up tweets yielding lower accuracy on the text classfier is considered a better text generator.

Sometimes running NLP systems end-to-end is very expensive, so another evaluation method, called intrinsic evaluation, can be used to quickly evaluate potential improvements of the model before using a more expensive extrinsic evaluation method. An intrinsic evaluation metric is one that measures the quality of a model independent of any application. For an instrinsic evaluation of the text generator we need a test set, sometimes called held out corpora that are not in our training sets. Therefore, to compare the performance of two text generators, we divide the data into training and test sets, train the parameters of both generators on the training set and then compare how well the two trained models fit the test set. By "fit the test set", we mean the probability that a language model (i.e., text generator) assigns to the test sentence. Whichever model assigns a higher probability to the test set - meaning it more accurately predicts the test set - is a better model.

However, in practice, perplexity is often used as the metric for evaluating language models instead of raw probability. The perplexity of a language model on a test set is the inverse probability of the test set, normalized by the number of words. It is related inversely to the likelihood of the test sequence according to the language model and can also be thought of as the weighted average branching factor of a language. Therefore, in comparing two text generators, whichever model achieves a lower preplexity on the test set - meaning it gives us more information about the word sequence - is a better model.

The performance of the text classifier is evaluated similarly to the intrinsic evalution of a language model. To compare the performance of two text classifiers, we also divide the data into training and test sets, train the parameters of both classifiers on the training set and then compare how well the two trained models fit the test set. By "fit the test set", we mean the ability to correctly classify the test sentence into its correct category (Trump or Not-Trump). Whichever model achieves a higher prediction accuracy on the test set - meaning it more accurately predicts the category of the test sentence - is a better model.

4 Related Work

There are primarily two types of language models for building text generators. One is statistical language models, which use traditional statistical techniques like N-grams, Hidden Markov Models (HMM) and certain linguistic rules to learn the probability distribution of words. Another is neural language models, which emerged recently and have surpassed the statistical language models in their effectiveness, using different kinds of neural networks to model language, including feedforward neural networks, recurrent neural networks and convolutional neural networks.

The underlying mathematics of the n-gram was first proposed by Markov (1913). Shannon (1948) applied n-grams to compute approximations to English word sequences. Based on Shannon's work, Markov models were commonly used in engineering, linguistic, and psychological work on modeling word sequences by the 1950s. Later, Jelinek (1976) at the IBM and Baker (1975) at CMU independently used n-grams in their speech recognition systems with some success. The highest accuracy language models by now are neural network language models, including feedforward language models (Bengio et al. 2006, Schewenk 2007), recurrent language models (Mikolov 2012), and gated convolutional networks language model (Dauphin 2016). They solve a major problem with n-gram language models: the number of parameters increase exponentially as the n-gram order increases, and n-grams have no way to generalize from training to test set. Neural language models

instead project words into a continuous space in which words with similar contexts have similar representations.

Add-one smoothing derives from Laplace's (1812) law of succession and was first applied as an engineering solution to the zero-frequency problem by Jeffreys (1948) based on an earlier add-k suggestion by Johnson (1932). Although add-k is useful for some tasks (including text classification), it doesn't work well for language modeling, generating counts with poor variances and often inappropriate discounts. Another smoothing method is backoff and interpolation. In backoff, we back off to a lower-order n-gram if we have zero evidence for a higher-order n-gram. For example, we use the trigram if the evidence is sufficient, otherwise we use the bigram, otherwise the unigram. In interpolation, we always mix the probability estimates from all the n-gram estimators, weighing and combining the trigram, bigram, and unigram counts. One of the most commonly used and best performing n-gram smoothing methods is the interpolated Kneser-Ney algorithm designed by Kneser and Ney (1995). Interpolated Kneser-Ney makes use of the probability of a word being a novel continuation and mixes a discounted probability with a lower-order continuation probability. The best-performing version of Kneser-Ney smoothing is called modified Kneser-Ney smoothing due to Chen and Goodman (1998).

There are also two primary approaches for building text classifiers. One is rule-based systems, which classify text into organized groups by using a set of handcrafted linguistic rules. These rules instruct the system to use semantically relevant elements of a text to identify relevant categories based on its content. Each rule consists of an antecedent or pattern and a predicted category. Rule-based systems are human comprehensible and can be improved over time, but these systems require deep knowledge of the domain. Also, generating rules for a complex system can be very time-consuming and these systems don't scale well given that adding new rules can affect the results of pre-exisiting rules. Another is machine learning based systems, which learns to make classifications based on past observations instead of relying on manually crafted rules. They are usually much more accurate than human-crafted rule systems, especially on complex classification tasks, and are also easier to maintain and scale by only feeding new tagged training examples to the model to learn new tasks. Some of the most popular machine learning algorithm for creating text classification models include traditional statistical methods like naive bayes (Wang & Manning 2012), support vector machine (Silva et al., 2011), modified logistic regression (Le & Mikolov, 2014), and neural network methods, including recursive neural networks (Wermter 2000) and convolutional neural networks (Kalchbrenner et al., 2014, Santos & Gatti, 2014, Shen et al., 2014).

5 Proposed Approach

There are two models to design in this project: a text generator for generating made-up tweets in the style of Trump, and a text classifier for distinguishing Trump's tweets from other people's tweets.

For the text generator, two different models are used and then compared. First, a traditional word n-gram model is used, including unigram, bigram, and trigram models, along with different smoothing methods applied to them, including add-k smoothing, backoff and interpolation, and kneser-ney smoothing. Second, a neural language model is used, including feedforward language models, recurrent language models, including simple recursive neural networks (RNN), long short-term memory (LSTM) and gated recurrent unit (GRU), and convolutional language model using gated convolutional neural networks (GCNN).

For the text classifier, various models, including naive bayes, support vector machine, bagging, CNN, RNN are used and compared. The classifier with the highest prediction accuracy on the test set is considered the best classifier and is in turn used to distinguish made-up tweets generated by the text generator from real Trump's tweets.

References

[1] Jurafsky, D., and Martin, J.H. (2019). N-gram Language Models. Speech and Language Processing (third edition draft).

[2] Shannon, C. E. (1948). A mathematical theory of communication. Bell System Technical Journal, 27(3), 379–423.

[3] Jelinek, F. (1976). Continuous speech recognition by statistical methods. Proceedings of the IEEE, 64(4), 532–557.

[4] Baker, J. K. (1975). The DRAGON system – An overview. IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-23(1), 24–29.

[5] Bengio, Y., Schwenk, H., Senecal, J.-S., Morin, F., and Gauvain, J.-L. (2006). Neural probabilistic language models. In Innovations in Machine Learning, 137–186. Springer.

[6] Schwenk, H. (2007). Continuous space language models. Computer Speech & Language, 21(3), 492–518.

[7] Mikolov, T. (2012). Statistical language models based on neural networks. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

[8] Dauphin, Y.N., Fan, A., Auli, M., and Grangier D. (2016). Language Modeling with Gated Convolutional Networks. Computation and Language. arXiv:1612.08083.

[9] Jeffreys, H. (1948). Theory of Probability (2nd Ed.). Clarendon Press. Section 3.23.

[10] Johnson, W. E. (1932). Probability: deductive and inductive problems (appendix to). Mind, 41(164), 421–423.

[11] Kneser, R. and Ney, H. (1995). Improved backing-off for Mgram language modeling. In ICASSP-95, Vol. 1, 181–184.

[12] Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Tech. rep. TR-10-98, Computer Science Group, Harvard University.

[13] Wang, S., and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2 (pp. 90-94). Association for Computational Linguistics.

[14] Silva, J., Coheur, L., Mendes, A. C., and Wichert, A. (2011). From symbolic to sub-symbolic information in question classification. Artificial Intelligence Review, 35(2), 137-154.

[15] Le, Q. V., and Mikolov, T. (2014). Distributed representations of sentences and documents. arXiv:1405.4053.

[16] Wermter, S. (2000). Neural network agents for learning semantic text classification. Information Retrieval, 3(2), 87-103.

[17] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. Acl, 655665.

[18] Santos, C. N. dos, and Gatti, M. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In COLING-2014 (pp. 6978).

[19] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management CIKM 14, 101110.

[20] Dickens, M. (2016). "I Have the Best Classifiers": Identifying Speech Imitating the Style of Donald Trump. CS224d Deep Learning for Natural Language Processing course project reports, Stanford University.