Duluth at SemEval-2017 Task 6: Language Models in Humor Detection

Xinru Yan & Ted Pedersen
Department of Computer Science
University of Minnesota Duluth
Duluth, MN, 55812 USA
{yanxx418,tpederse}@d.umn.edu

Abstract

This paper describes the Duluth system that participated in SemEval-2017 Task 6 #HashtagWars: Learning a Sense of Humor. The system participated in Subtasks A and B using N-gram language models, ranking highly in the task evaluation. This paper discusses the results of our system in the development and evaluation stages and from two post-evaluation runs.

1 Introduction

Humor is an expression of human uniqueness and intelligence and has drawn attention in diverse areas such as linguistics, psychology, philosophy and computer science. *Computational humor* draws from all of these fields and is a relatively new area of study. There is some history of systems that are able to generate humor (e.g., (**?**), (**?**)). However, *humor detection* remains a less explored and challenging problem (e.g., (**?**), (**?**), (**?**), (**?**)).

SemEval-2017 Task 6 (**?**) also focuses on *humor detection* by asking participants to develop systems that learn a sense of humor from the Comedy Central TV show, @*midnight with Chris Hardwick*. Our system ranks tweets according to how funny they are by training N-gram language models on two different corpora. One consisting of funny tweets provided by the task organizers, and the other on a freely available research corpus of news data. The funny tweet data is made up of tweets that are intended to be humorous responses to a hashtag given by host Chris Hardwick during the program.

2 Background

Training **Language Models** (LMs) is a straightforward way to collect a set of rules by utilizing the fact that words do not appear in an arbitrary order; we in fact can gain useful information about a word by knowing the company it keeps (**?**). A statistical language model estimates the probability of a sequence of words or an upcoming word. An N-gram is a contiguous sequence of N words: a unigram is a single word, a bigram is a two-word sequence, and a trigram is a three-word sequence. For example, in the tweet

tears in Ramen #SingleLifeIn3Words

"tears", "in", "Ramen" and "#SingleLifeIn3Words" are unigrams; "tears in", "in Ramen" and "Ramen #SingleLifeIn3Words" are bigrams and "tears in Ramen" and "in Ramen #SingleLifeIn3Words" are trigrams.

An N-gram model can predict the next word from a sequence of N-1 previous words. A trigram Language Model (LM) predicts the conditional probability of the next word using the following approximation:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-2}, w_{n-1}) \quad (1)$$

The assumption that the probability of a word depends only on a small number of previous words is called a **Markov** assumption (**?**). Given this assumption the probability of a sentence can be estimated as follows:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-2}, w_{k-1}) \quad (2)$$

In a study on how phrasing affects memorability, (**?**) take a language model approach to measure the distinctiveness of memorable movie quotes. They do this by evaluating a quote with respect to a "common language" model built from the newswire sections of the Brown corpus (**?**). They find that movie quotes which are less like "common language" are more distinctive and

therefore more memorable. The intuition behind our approach is that humor should in some way be memorable or distinct, and so tweets that diverge from a "common language" model would be expected to be funnier.

In order to evaluate how funny a tweet is, we train language models on two datasets: the tweet data and the news data. Tweets that are more probable according to the tweet data language model are ranked as being funnier. However, tweets that have a lower probability according to the news language model are considered the funnier since they are the least like the (unfunny) news corpus. We relied on both bigrams and trigrams when training our models.

We use KenLM (**?**) as our language modeling tool. Language models are estimated using modified Kneser-Ney smoothing without pruning. KenLM also implements a back-off technique so if an N-gram is not found, KenLM applies the lower order N-gram's probability along with its back-off weights.

3 Method

Our system[1] estimated tweet probability using N-gram LMs. Specifically, it solved the comparison (Subtask A) and semi-ranking (Subtask B) subtasks in four steps:

1. Corpus preparation and pre-processing: Collected all training data into a single file. Preprocessing included filtering and tokenization.

2. Language model training: Built N-gram language using KenLM.

3. Tweet scoring: Computed log probability for each tweet based on trained N-gram language model.

4. Tweet prediction: Based on the log probability scores.

    - Subtask A – Given two tweets, compare and predict which one is funnier.
    - Subtask B – Given a set of tweets associated with one hashtag, rank tweets from the funniest to the least funny.

[1] https://xinru1414.github.io/HumorDetection-SemEval2017-Task6/

3.1 Corpus Preparation and Pre-processing

The tweet data was provided by the task organizers. It consists of 106 hashtag files made up of about 21,000 tokens. The hashtag files were further divided into a development set *trial_dir* of 6 hashtags and a training set of 100 hashtags *train_dir*. We also obtained 6.2 GB of English news data with about two million tokens from the News Commentary Corpus and the News Crawl Corpus from 2008, 2010 and 2011[2]. Each tweet and each sentence from the news data is found on a single line in their respective files.

3.1.1 Preparation

During the development of our system we trained our language models solely on the 100 hashtag files from *train_dir* and then evaluated our performance on the 6 hashtag files found in *trial_dir*. That data was formatted such that each tweet was found on a single line.

3.1.2 Pre-processing

Pre-processing consists of two steps: filtering and tokenization. The filtering step was only for the tweet training corpus. We experimented with various filtering and tokenziation combinations during the development stage to determine the best setting.

- Filtering removes the following elements from the tweets: URLs, tokens starting with the "@" symbol (Twitter user names), and tokens starting with the "#" symbol (Hashtags).

- Tokenization: Text in all training data was split on white space and punctuation

3.2 Language Model Training

Once we had the corpora ready, we used the KenLM Toolkit to train the N-gram language models on each corpus. We trained using both bigrams and trigrams on the tweet and news data. Our language models accounted for unknown words and were built both with and without considering sentence or tweet boundaries.

3.3 Tweet Scoring

After training the N-gram language models, the next step was scoring. For each hashtag file that needed to be evaluated, the logarithm of the probability was assigned to each tweet in the hashtag file

[2] http://www.statmt.org/wmt11/featured-translation-task.html

based on the trained language model. The larger the probability, the more likely that tweet was according to the language model. Table 1 shows an example of two scored tweets from hashtag file *Bad_Job_In_5_Words.tsv* based on the tweet data trigram language model. Note that KenLM reports the log of the probability of the N-grams rather than the actual probabilities so the value closer to 0 (-19) has the higher probability and is associated with the tweet judged to be funnier.

3.4 Tweet Prediction

The system sorts all the tweets for each hashtag and orders them based on their log probability score, where the funniest tweet should be listed first. If the scores are based on the tweet language model then they are sorted in ascending order since the log probability value closest to 0 indicates the tweet that is most like the (funny) tweets model. However, if the log probability scores are based on the news data then they are sorted in descending order since the largest value will have the smallest probability associated with it and is therefore least like the (unfunny) news model.

For Subtask A, the system goes through the sorted list of tweets in a hashtag file and compares each pair of tweets. For each pair, if the first tweet was funnier than the second, the system would output the tweet_ids for the pair followed by a "1". If the second tweet is funnier it outputs the tweet_ids followed by a "0". For Subtask B, the system outputs all the tweet_ids for a hashtag file starting from the funniest.

4 Experiments and Results

In this section we present the results from our development stage (*Table 2*), the evaluation stage (*Table 3*), and two post-evaluation results (*Table 3*). Since we implemented both bigram and trigam language models during the development stage but only results from trigram language models were submitted to the task, we evaluated bigram language models in the post-evaluation stage. Note that the accuracy and distance measurements listed in Table 2 and Table 3 are defined by the task organizers (**?**).

Table 2 shows results from the development stage. These results show that for the tweet data the best setting is to keep the # and @, omit sentence boundaries, be case sensitive, and ignore tokenization. While using these settings the trigram language model performed better on Subtask B (.887) and the bigram language model performed better on Subtask A (.548). We decided to rely on trigram language models for the task evaluation since the advantage of bigrams on Subtask A was very slight (.548 versus .543). For the news data, we found that the best setting was to perform tokenization, omit sentence boundaries, and to be case sensitive. Given that trigrams performed most effectively in the development stage, we decided to use those during the evaluation.

Table 3 shows the results of our system during the task evaluation. We submitted two runs, one with a trigram language model trained on the tweet data, and another with a trigram language model trained on the news data. In addition, after the evaluation was concluded we also decided to run the bigram language models as well. Contrary to what we observed in the development data, the bigram language model actually performed somewhat better than the trigram language model. In addition, and also contrary to what we observed with the development data, the news data proved generally more effective in the post–evaluation runs than the tweet data.

5 Discussion and Future Work

We relied on bigram and trigram language models because tweets are short and concise, and often only consist of just a few words.

The performance of our system was not consistent when comparing the development to the evaluation results. During development language models trained on the tweet data performed better. However during the evaluation and post-evaluation stage, language models trained on the news data were significantly more effective. We also observed that bigram language models performed slightly better than trigram models on the evaluation data. This suggests that going forward we should also consider both the use of unigram and character–level language models.

These results suggest that there are only slight differences between bigram and trigram models, and that the type and quantity of corpora used to train the models is what really determines the results.

The task description paper (**?**) reported system by system results for each hashtag. We were surprised to find that our performance on the hashtag file *#BreakUpIn5Words* in the evaluation stage

The hashtag: #BadJobIn5Words		
tweet_id	tweet	score
705511149970726912	The host of Singled Out #BadJobIn5Words @midnight	-19.923433303833008
705538894415003648	Donut receipt maker and sorter #BadJobIn5Words @midnight	-27.67446517944336

Table 1: Scored tweets according to the trigram LM. The log probability scores computed based on the trigram LM are shown in the third column.

DataSet	N-gram	# and @ removed	Sentence Boundaries	Lowercase	Tokenization	Subtask A Accuracy	Subtask B Distance
tweets	**trigram**	**False**	**False**	**False**	**False**	**0.543**	**0.887**
tweets	bigram	False	False	False	False	0.548	0.900
tweets	trigram	False	True	True	False	0.522	0.900
tweets	bigram	False	True	True	False	0.534	0.887
news	**trigram**	**NA**	**False**	**False**	**True**	**0.539**	**0.923**
news	bigram	NA	False	False	True	0.524	0.924
news	trigram	NA	False	False	False	0.460	0.923
news	bigram	NA	False	False	False	0.470	0.900

Table 2: Development results based on *trial_dir* data. The settings we chose to train LMs are in bold.

DataSet	N-gram	# and @ removed	Sentence Boundaries	Lowercase	Tokenization	Subtask A Accuracy	Subtask B Distance
tweets	**trigram**	**False**	**False**	**False**	**False**	**0.397**	**0.967**
tweets	bigram	False	False	False	False	0.406	0.944
news	**trigram**	**NA**	**False**	**False**	**True**	**0.627**	**0.872**
news	bigram	NA	False	False	True	0.624	0.853

Table 3: Evaluation results (bold) and post-evaluation results based on *evaluation_dir* data. The trigram LM trained on the news data ranked 4th place on Subtask A and 1st place on Subtask B.

was significantly better than any other system on both Subtask A (with accuracy of 0.913) and Subtask B (with distance score of 0.636). While we still do not fully understand the cause of these results, there is clearly something about the language used in this hashtag that is distinct from the other hashtags, and is somehow better represented or captured by a language model. Reaching a better understanding of this result is a high priority for future work.

The tweet data was significantly smaller than the news data, and so certainly we believe that this was a factor in the performance during the evaluation stage, where the models built from the news data were significantly more effective. Going forward we plan to collect more tweet data, particularly those that participate in #HashTagWars. We also intend to do some experiments where we cut the amount of news data and then build models to see how those compare.

While our language models performed well, there is some evidence that neural network models can outperform standard back-off N-gram models (**?**). We would like to experiment with deep learning methods such as recurrent neural networks, since these networks are capable of forming short term memory and may be better suited for dealing with sequence data.

References

Cristian Danescu-Niculescu-Mizil, Justin Cheng, Jon Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. In *Proceedings of the 50th Annual Meeting of the Associ-*

ation for Computational Linguistics: Long Papers - Volume 1. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '12, pages 892–901.

J. Firth. 1968. A synopsis of linguistic theory 1930-1955. In F. Palmer, editor, *Selected Papers of J. R. Firth*, Longman.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 690–696.

Henry Kucera and W. Nelson Francis. 1967. *Computational Analysis of Present-day American English*. Brown University Press, Providence, RI, USA.

A. A. Markov. 2006. An example of statistical investigation of the text *Eugene Onegin* concerning the connection of samples in chains. *Science in Context* 19(4):591–600.

Rada Mihalcea and Carlo Strapparava. 2006. Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence* 22(2):126–142.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, pages 5528–5531.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of English puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 719–729.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 703–711.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. SemEval-2017 Task 6: #HashtagWars: learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, BC.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '15, pages 1065–1074.

Oliviero Stock and Carlo Strapparava. 2003. Getting serious about the development of computational humor. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. Acapulco, pages 59–64.

Renxian Zhang and Naishi Liu. 2014. Recognizing humor on Twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '14, pages 889–898.