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

Xinru Yan
Department of Computer Science
University of Minnesota Duluth
Duluth, MN, 55812 USA
yanxx418@d.umn.edu

Ted Pedersen
Department of Computer Science
University of Minnesota Duluth
Duluth, MN, 55812 USA
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 Subtask A and Subtask B using N-gram language models, ranking highly in the evaluation. This paper includes the results of our system during development and evaluation stage and two post-evaluation results.

1 Introduction

Humor represents human uniqueness and intelligence to some extent and has continuously drawn attention in different research areas such as linguistics, psychology, philosophy and computer science. In computer science, relevant theories derived from those fields have formed a relatively new area of study, *computational humor* (Zhang and Liu, 2014). Humor has not been addressed broadly in current computational research. Many studies have developed effective systems to produce humor (Özbal and Strapparava, 2012). However, *humor detection* is essentially a more challenging and fun problem. For example, Mihalcea and Strapparava draw particular attention on automatic humor recognition in their work (Mihalcea and Strapparava, 2006). SemEval-2017 Task 6 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 detects humor by training N-gram language models on two sets of training data, the tweets data and the news data.

2 Background

Training **Language Models** (LMs) is a straightforward way to collect set of rules by utilizing the fact that words do not appear in an arbitrary order, which means we can gain useful information from a word and its neighbors (Jurafsky and Martin, 2009). A statistical language model is a model that computes the probability of a sequence of words or an upcoming word (Jurafsky and Martin, 2009).

The idea of word prediction with probabilistic models is called the N-gram model, which predicts the upcoming word from the previous N-1 words. An N-gram is a contiguous sequence of N words: a unigram is a single word, a bigram is a two-word sequence of words and a trigram is a three-word sequence of words. For example, in 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.

When we use for instance, a trigram Language Model (LM), to predict the conditional probability of the next word, we are thus making the following approximation:

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

The assumption that the probability of a word depends only on a small number of previous words is called the **Markov** assumption (Markov, 2006). According to the Markov assumption, here we show the general equation for computing the probability of a complete word sequence using a trigram LM:

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

In the study on how phrasing affects memorability, in order to analyze the characteristics of memorable quotes, researchers take the language model approach to investigate the distinctiveness feature (Danescu-Niculescu-Mizil et al., 2012).

Specifically, to evaluate how distinctive a quote is, they evaluate its likelihood with the respect of the "common language" model which consists of the newswire sections of the Brown corpus (Kucera and Francis., 1967). They employ LMs on the "common language" and come to the conclusion that movie quotes which are less like the "common language" (i.e., more distinctive) are more memorable. The idea of using LMs to assess the memorability of a quote based on its distictiveness is suitable for our purpose of detecting how humorous (i.e., how distinctive a tweet is comparing to other tweets). In our case, tweets that are less like the "common language" (i.e., more distinctive) are considered to be funnier. We use English news as our "common language".

In order to evaluate how funny a tweet is, we train LMs on two datasets: the tweets data and the news data. For the tweets data LMs, tweets that are more like the tweets LMs (i.e., have higher probability score) are ranked as being more funny. For the news data LMs, tweets that are less like the news LMs (i.e., have lower probability score) are ranked as being more funny. For our purpose, we train bigram LMs and trigram LMs on both sets of training data.

KenLM, as a language modeling tool, is used in our system (Heafield et al., 2013). LMs are estimated from the corpus using modified Kneser-Ney smoothing without pruning. KenLM reads a text file and generates LMs in ARPA format. KenLM also implements back-off technique, which means if the N-gram is not found, it applies the lower order N-gram's probability along with its back-off weights. Instead of using the real probability of the N-gram, KenLM applies the logarithm scheme.

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 files to form one training corpus. Pre-processing included filtering and tokenization.

2. Language model training: Built N-gram LMs by feeding the corpus to the KenLM Language Model Toolkit.

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

4. Tweet prediction: Based on the log probability
   - Subtask A – Given two tweets, compared them and predicted which one is funnier.
   - Subtask B – Given a set of tweets associated with one hashtag, ranked tweets from the funnest to the least funny.

3.1 Corpus Preparation and Pre-processing

The Duluth system uses two distinct sets of training data: the tweets data and the news data. The tweets data was provided by the SemEval task. It consisted of 106 hashtag files with about 21,000 tokens. We collected in total of 6.2 GB of English news data with about 2,000,000 tokens from the News Commentary Corpus and the News Crawl Corpus from 2008, 2010 and 2011 [2].

3.1.1 Preparation

To prepare the tweets training corpus, the system took each tweet from the hashtag files, which included tweets from both *train_dir* and *trial_dir* from the task, and created a text file with each tweet on its own line. During the development stage of the system we trained LMs solely on the *train_dir* data, which included 100 hashtag files; we tested the system on the *trial_dir* data consisting of 6 hashtag files. For the news data, the system read each sentence from the news files and created a text file with each sentence per line to form the news training corpus.

3.1.2 Pre-processing

The pre-processing consisted 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: the filtering process included removing the following elements from the tweets:

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

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

    - URLs
    - Twitter user names: Tokens starting with the “@” symbol
    - Hashtags: Tokens starting with the “#” symbol

- Tokenization: For both training data sets we split text by spaces and punctuation marks

3.2 Language Model Training

Once we had the corpora ready, we used the KenLM Toolkit to train the N-gram LMs on each corpus. We trained two different LMs, bigrams and trigrams, on both tweets and news training data sets. Our LMs were trained to able to handle the unknown words to estimate the probability of a tweet more accurately. To derive the best setting of the LMs for both tasks, we experimented using the language model with and without sentence boundaries.

3.3 Tweet Scoring

After training the N-gram LMs, the next step was scoring. For each hashtag file that needed to be evaluated, based on the trained N-gram LM, a logarithm score was assigned by our system for each tweet in the hashtag file. The larger the score, the more likely that the tweet appeared with respect to that LM. Table 1 shows an example of two scored tweets from hashtag file *Bad_Job_In_5_Words.tsv* based on the trigram LM trained on the tweets data.

3.4 Tweet Prediction

The system sorted tweets for each hashtag file based on their score, meaning the funniest one was listed on the top (i.e., if the system used a tweets LM), the tweets would be sorted in descending order. In the case that it used a news LM, the tweets would be sorted in ascending order. For Subtask A, given a hashtag file, the system went through the sorted list of tweets, compared each pair of tweets and produced a tsv format file. For each tweet pair, if the first tweet was funnier than the second one, the system would output the tweet_ids for the pair followed by “1”. Otherwise it output the tweet_ids followed by “0”. For Subtask B, given a hashtag file, the system output the tweet_ids starting from the funniest.

4 Experiments and Results

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

Table 2 shows results from the development stage. From this table we can estimate the best setting to train LMs for both data sets. For tweets data, keeping the # and @, omitting sentence boundary, being case sensitive and ignoring tokenization is the best setting. Under the best setting, trigram LM performed better on Subtask B and bigram LM performed better on Subtask A. We decided to train trigram LMs since the advantage of trigram LMs on Subtask B is relatively more significant. For the news data, tokenization helped to improve the performance of LMs. Omitting sentence boundaries, being case sensitive along with tokenization is the best setting. Trigram performed better considering both tasks under the best setting so we chose to train trigram LMs.

Table 3 shows the results of our system applying trigram LMs during evaluation along with bigram LMs results from the post-evaluation runs. For tweets data, bigram LM performed slightly better on both tasks than trigram LM. For news data, trigram LM had a slight advantage over bigram LM on Subtask A and bigram LM performed better on Subtask B.

5 Discussion and Future Work

We focused on training bigram and trigram LMs because tweets are normally short and concise. During the development stage, LMs trained on the tweets data performed better than LMs trained on the news data. However during the evaluation and post-evaluation stage, LMs trained on the news data had a significant improvement. We believe there are two reasons. Firstly, the news data LMs had a huge success on one of the hashtag files, *BreakUpIn5Words*, on both Subtask A (with accuracy of 0.913) and Subtask B (with distance score at 0.636) in evaluation_dir. Secondly, the trial_dir data is more similar to the train_dir data than the evaluation_dir data, which could cause the tweets

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.

LMs trained on the train_dir data capturing characteristics better on the trial_dir than on the evaluation_dir. The results also showed that generally speaking when using the same training dataset, bigram LMs had a slightly better performance than trigram LMs, although their performances are really similar. This indicates that for this task there are no significant differences between using bigram LMs and trigram LMs. It also suggests that in the future we could lightly focus on bigram LMs and even try unigram LMs and character-level LMs.

After comparing the amount of tweets data and news data we used, we believe that the lack of tweets data could have caused the tweets LMs to perform worse. Therefore, one way to improve the system, especially the tweets data LM, is to collect more tweets that participate in the hashtag wars. We would also like to train news LMs using the same amount of data we have for the tweets to see how the results compare. Additionally, we want to gather more news data and see if the quantity of news data would still make a difference.

Furthermore, we would like to try some machine learning techniques, specifically deep learning methods such as recurrent neural networks. Studies have shown that neural network based LMs work effectively and outperform standard back-off N-gram models (Mikolov et al., 2011). In addition, recurrent neural networks are capable of forming short term memory so it can better deal with problems associated with sequences. It would be interesting to see if some combination of these methods could enhance the system.

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 Association for Computational Linguistics: Long Papers - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '12, pages 892–901. http://dl.acm.org/citation.cfm?id=2390524.2390647.

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.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

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):591600. https://doi.org/10.1017/S0269889706001074.

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.

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.

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. https://doi.org/10.1145/2661829.2661997.