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 completed Subtask A and Subtask B using N-gram language models, ranking well during evaluation. This paper includes the evaluation results of our system with several post-evaluation runs.

1 Introduction

Since humor represents human uniqueness and intelligence to some extent, it 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 young area of study - computational humor (Zhang and Liu, 2014). Humor has not been addressed broadly in current computational research. Many studies have developed decent systems to produce humor (Ozbal 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, Duluth, applies language model approach to detect humor by training N-gram language models on two sets of training data, the tweets data and the news data.

2 Background

Language models(LMs) are 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 some 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). Below are two examples of language modeling:

To compute the probability of a sequence of words W given the sequence (w_1, w_2, w_3), we have:

$$P(W) = P(w_1, w_2, ...w_n) \quad (1)$$

To compute the probability of an upcoming word $w3$ given the sequence (w_1, w_2), the language model gives us the following probability:

$$P(w_3|w_1, w_2) \quad (2)$$

The idea of word prediction with probabilistic models is called N-gram models, which predict 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 example, trigram LM, to predict the conditional probability of the next word, we are thus making the following approximation:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-2}, w_{n-1}) \quad (3)$$

The assumption that the probability of a word

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

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-2}, w_{k-1}) \qquad (4)$$

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 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. They employ six add-1 smoothed LMs – unigram, bigram, trigram language models and unigram, bigram, trigram Part of Speech language models – on the "common language". They come to the conclusion that movie quotes which are less like the "common language" are more memorable. The idea of using language models to assess the memorability of a quote is suitable for our purpose of detecting how humorous a tweet is. Except for using tweets provided by the task to train N-gram LMs, our system also trained N-gram LMs on English news data in order to evaluate how distinctive, in this case, how funny, a tweet is comparing to the "common language"– news. Tweets that were more like the tweets model, or less like the news model, were ranked as being more funny. For our purpose, we trained bigram LMs and trigram LMs on both sets of training data.

3 Method

Our system estimates tweet probability using N-gram LMs. Specifically, it solves the given two subtasks in four steps:

1. Corpus preparing and pre-processing: Collect all training data files to form one training corpus. Pre-processing includes filtering and tokenization.

2. Language model training: Build N-gram LMs by feeding the corpus to KenLM Language Model Toolkit (Heafield et al., 2013).

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

4. Tweet prediction: According to the log probability

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

3.1 Corpus preparing and pre-processing

In our system, we use two distinct sets of training data: the tweets data and the news data. The tweets data is provided by the SemEval task. It consists of 106 hashtag files with about 21,000 tokens. In addition, 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 [1].

3.1.1 Preparing

To prepare the tweets training corpus, the system took each tweet from the hashtag files, which includes tweets from both *trial_dir* and *train_dir* from the task, and creates a file with each tweet on its own line. Be aware that during the development phase of the system, we trained LMs solely on the *trial_dir* data, which includes 100 hashtag files, and tested it on the *train_dir* data consisting of 6 hashtag files. For the news data, the system read each sentence from the news files and create a file with each sentence per line to form the news training corpus.

3.1.2 Pre-processing

In general, the pre-processing consists of two steps: filtering and tokenization. The filtering step was only for the tweet training corpus. Also, we experimented various filtering and tokenziation combinations during development stage to determine the best settings.

- Filtering: the filtering process includes removing the following elements from tweets:
  - 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

[1] http://www.statmt.org/wmt11/translation-task.html#download

3.2 Language Model Training

Once we have the corpora ready, we use the KenLM Toolkit to train the N-gram LMs on each corpus. LMs are estimated from the corpus using modified Kneser-Ney smoothing without pruning. KenLM reads a plain text file and generates LMs in arpa format. We trained two different language models – bigrams and trigrams – ob both tweets and news training data sets. KenLM also implements back-off technique, which simply 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 a base 10 logarithm scheme. Table 1 shows an example arpa file of the trigram LM we trained on the tweets data:

N-gram 1=21580 N-gram 2=60624 N-gram 3=73837
unigram: -4.8225346 <unk> 0 0 <s> -0.47505832 -1.4503417 </s> 0 -4.415446 Donner -0.12937292 ...
bigrams: ... -0.9799023 Drilling Gulf -0.024524588 ...
trigrams: ... -1.171928 I'll start thinking ...

Table 1: Trigram LM on tweets data

Each N-gram line starts with the base 10 logarithm probability of that N-gram, followed by the N-gram which consists of N words. The base 10 logarithm of the back-off weight for the N-gram optionally follows after. In the trigram LM, there are 73,837 trigrams in total from the tweet training corpus. Notice that there are three "special" words in a language model: the beginning of a sentence denoted by <s>, the end of a sentence denoted by </s> and the out of vocabulary word denoted by <unk>. In order to be able to handle the unknown words to estimate the probability of a tweet more accurately, in all our experiments we kept the <unk> word in our LMs. To derive the best setting of language model 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 is scoring. For each hashtag file that needs to be evaluated, based on the trained N-gram LM, our system assigns a base 10 log probability as a score for each tweet in the hashtag file. The larger the score, the more likely that the tweet appears with respect to that LM. Table 2 shows an example of scored two 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 sorts tweets for each hashtag file based on their score, meaning the funniest one is listed on the top i.e. if the system uses a tweets LM, the tweets are sorted in descending order. In the case that it uses a news LM, the tweets are sorted in ascending order. For Subtask A, given a hashtag file, the system goes through the sorted list of tweets, compares each pair of tweets and produces a tsv format file as the task asks for. For each tweet pair, if the first tweet is funnier than the second one, system outputs the tweet_ids for the pair followed by "1". Otherwise it outputs the tweet_ids followed by "0". For Subtask B, given a hashtag file, the system simply outputs the tweet_ids starting from the funniest.

4 Experiments and Results

In this section we present the evaluation results of our system, as well as several post-evaluation experiments. Since we implemented both bigram and trigam LMs during the development stage but only results from trigram LMs were submitted for the task, we evaluated bigram LMs in post-evaluation stage as well.

Table 3 shows results from developing stage. Note that for tweets data we trained language models on *train_dir* data and tested on *trial_dir* data. From this table we can estimate the best setting to train language models for both data sets: for tweets data we decided to use trigrams and omit sentence boundaries; for news data we chose to train trigram LMs on a tokenized news corpus.

Table 4 shows the results of our system applying trigram LMs during evaluation along with bigram LMs results from the post-evaluation runs.

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 2: Scored tweet according to trigram LM. The format follows .tsv file provided by the task. The first column shows tweets_id; the second column shows tweets; the third column shows the probability score computed based on the trigram LM.

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

Table 3: Development results. The accuracy and distance measurements are provided by the task. In general, trigram LMs outperform bigram LMs.

DataSet	N-gram	# & @ 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 4: Evaluation results and post-evaluation runs. Trigram LM trained on news data ranked 4th place for Subtask A and 1st place for Subtask B during evaluation.

It demonstrates that generally trigram LMs work better than bigram LMs.

5 Discussion and Future Work

We focused on training bigram and trigram LMs considering tweets are normally short and concise. Trigrams outperform bigrams considering trigrams have relatively better coverage than bigrams.

After comparing the amount of tweets data and news data we used, we believe that 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 about 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.

Besides, we would also like to try some machine learning techniques, specifically deep learning method 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 networks are capable of forming short term memory so it can better deal with problems associated with sequences. From our perspective, deep learning method could play a role in this task. It would also 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.

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 Ozbal and Carlo Strapparava. 2012. Computational humour for creative naming. *Computational Humor 2012* page 15.

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.