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 Task A and Task B using ngram language models, ranking well during evaluation. This paper includes the 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 (?). Humor has not been addressed broadly in current computational research. Many studies have developed decent systems to produce humor (?). However, humor detection is essentially a more challenging and fun task. 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 applies language model approach to detect humor by training ngram models on two sets of training data, the tweets data and the news data.

2 Background

Language models 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 learn a lot from a word and its neighbors (?). A statistical language model is a model that computes the probability of a sequence of words or an upcoming word (?).
Below are two examples of language modeling: To compute the probability of a sequence of words W given the sequence $(w_1, w_2, ...w_n)$, 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. Here we show the general equation for computing the probability of a complete word sequence using trigram language model:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-2}, w_{k-1}) \quad (3)$$

In the study on how phrasing affects memorability, in order to analyze the characteristics of memorable quotes, researchers take language model approach to investigate distinctiveness feature and employ syntactic measures on the data to evaluate generality feature (?). Specifically, in favor of evaluating 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 language modelsunigram, bigram, trigram word language models and unigram, bigram, trigram Part of Speech (POS) language modelson the common languagemodel. The idea of using language models to assess the memorability of a quote is suitable for our purpose of detecting how humorous a twitter is. Except for using funny tweets provided by the task to train ngram language models, our system also trained ngram language models on English news data in order to evaluate how distictive, in this case, how funny, a tweet is comparing to news. For our purpose, we trained unigrams, bigrams and trigrams on both sets of training data.

3 Method

Our system estimates tweet probability using ngram models. Specifically, it solves the given problem 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 language models by feeding the corpus to KenLM Language Model Toolkit (**?**).

3. Tweet scoring: Get log probability for each tweet based on the trained ngram language model.

4. Tweet prediction: According to the log probability
    - Given two tweets, comparing two tweets and predicting which one is funnier (for subtask A)
    - Given a set of tweets associated with one specific hashtag, ranking tweets from the funnest to the least funny (for subtask B)

3.1 Corpus preparing and pre-processing

In our system, we used 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, about 21580 tokens. In addition, we collected in total of 6.2 GB of English news data, about 2002655 tokens, from the News Commentary Corpus and the News Craw Corpus from yeas of 2008, 2010 and 2011 [1].

3.1.1 Preparing

To prepare the tweets data, the system takes in total of 106 hashtag files, which includes both trial_dir and train_dir from the task, and put all tweets in one plain text file to form the tweet training corpus. Each tweet is on its own line. Be aware that during the development phase of the system, we trained the language model on the train_dir data and tested it on the trial_dir data.
For the news data, the system reads in all the sentences from the news files and again, put them in one plain text file to form the news training corpus. Each sentence takes its own line.

3.1.2 Pre-processing

In general, the pre-processing consists of two steps: filtering and tokenization. The filtering step is mainly for the tweet training corpus. Also, we applied various filtering and tokenziation combinations on experiments to determine the best settings (see section 4).

- Filtering: the filtering process includes removing following elements from the :
    - URLs
    - Twitter user names with symbol @ indicating the user name
    - Hashtags with symbol # indicating the topic of the tweet
- Tokenization: For both training data sets we splitted text by space and punctuation marks

3.2 Language Model Training

Once we have the corpus ready, we use the KenLM Toolkit to train the n-gram language models on the corpus. Language models are estimated from the corpus using modified Kneser-Ney smoothing without pruning. KenLM reads in a plain text file and generates language models in arpa format. We trained three different language models – unigrams, bigrams and trigrams – for both training data sets. KenLM also implements back off technique, which simply means it applies the lower order ngram's probability along with its back-off weights if the ngram is not found. Instead of using the real probability of the ngram, KenLM applies

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

base 10 logarithm scheme. Here is an example of the trigram model we trained on the tweets data:

ngram 1=21580 ngram 2=60624 ngram 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 model on tweets data

Each ngram line starts with the base 10 logarithm probability of that ngram, followed by the ngram which consists of n words. The base 10 logarithm of the back-off weight for the ngram is followed after optionally. In this trigram language model trained on the tweets data, there are 73837 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 keep the <unk>word in the language model. To figure out the best setting of language model for both tasks, we experiment using the language model with and without sentence boundaries.

3.3 Tweet Scoring

After training the ngram model, the next step is scoring. For each hashtag file that needs to be evaluated, based on the trained ngram language model, the system assigns a base 10 log probability for each tweet in the hashtag file. Here is an example of scored tweet from hashtag file Bad_Job_In_5_Words.tsv based on the ngram language model trained on the tweets data:
705511149970726912 The host of Singled Out #BadJobIn5Words @midnight -19.923433303833008 705538894415003648 Donut receipt maker and sorter #BadJobIn5Words @midnight -27.67446517944336

3.4 Tweet Prediction

The system sorts tweets for each hashtag file based on their score in descending order, meaning the most probable one is listed on the top. For Task A, given a hashtag file, the system goes through the sorted list of tweets, compare each pair of tweets and produces a tsv format file as the task asks for. For each tweet pair twee_1 and tweet_2, if tweet_1 has higher score, system outputs tweet_ids for the pair followed by ”1” and followed by ”0” otherwise. For Task B, given a hashtag file, the systm simply outputs tweet_ids in the order of the sorted list.

4 Experiments and Results

In this section we present the evaluation results of our system, as well as several post-evaluation experiments. Notice the system used trigram language model in evaluation. Bigram language models were used during system development and post-evaluation.

Table x.x 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 tell the best setting to train language models for both data sets: for tweets data we decided to use trigrams and omit sentence boundaries (in this case, tweet boundaries); for news data we chose to train trigram language models on a tokenized news corpus.

DS	N	TS	SB	LC	TK	AA	BD
t	3	T	F	F	F	0.543	0.887
t	3	T	T	T	F	0.522	0.900
t	2	T	F	F	F	0.548	0.900
n	3	NA	F	F	T	0.539	0.923
n	3	NA	F	F	F	0.460	0.923
n	2	NA	F	F	F	0.470	0.900

Table 2: Development results

sb stands for sentence boundaries

Since in developing stage we implemented bigram and trigram language models, we added bigram language models in post-evaluation stage. Table x.x shows the results of our system applying trigram models during evaluation along with bigram model results:

DS	N	TS	SB	LC	TK	AA	BD
t	3	T	F	F	F	0.397	0.967
t	2	T	F	F	F	0.406	0.944
n	3	NA	F	F	T	0.627	0.872
n	2	NA	F	F	T	0.624	0.853

Table 3: Evaluation results and post-evaluation runs

5 Discussion and Future Work

We believe that lack of tweets data could cause the failure on the language models comparing the amount of tweets data and news data we used. One way to improve the system, especially the tweets data language model, is to collect more tweets that participate in the hashtag wars. Additionally, we want to try to gather more news data and see if quantity of news training data would still make a difference. Besides applying ngram language model approach to the task, we would also like to try some machine learning techniques, specifically deep learning method such as word2vec. From our perspective, deep learning method could play a role in this task in the sense of developing a system that learns humor from the show's point of view through neurons. It would also be interesting to see if even combining these two methods could help enhance the system.

References

Cristian Danescu-Niculescu-Mizil, Justin Cheng, and Jonand Lee Lillian Kleinberg. 2012. You had me at hello: How phrasing affects memorability. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 892–901. http://aclweb.org/anthology/P12-1094.

Daniël De Kok and Harm Brouwer. 2011. Natural language processing for the working programmer. Del.

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. http://kheafield.com/professional/edinburgh/estimate_paper.pdf.

James H Martin and Daniel Jurafsky. 2000. Speech and language processing. volume 710.

Gözde Ozbal and Carlo Strapparava. 2012. Computational humour for creative naming. 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, pages 889–898.