Optimizing Language Models for Polarity Classification

Michael Wiegand and Dietrich Klakow

Spoken Language Systems, Saarland University, Germany
{Michael.Wiegand|Dietrich.Klakow}@lsv.uni-saarland.de

Abstract. This paper investigates the usage of various types of language models on polarity text classification – a subtask in opinion mining which deals with distinguishing between positive and negative opinions in natural language. We focus on the intrinsic benefit of different types of language models. This means that we try to find the optimal settings of a language model by examining different types of normalization, their interaction with smoothing and the benefit of class-based modeling.

1 Introduction

There has been an increasing interest in opinion mining in recent years, in particular, in polarity text classification. Though Bayesian methods have been widely explored in this context, for example in [1], less attention has been drawn to the impact of language modeling on this classification task. This paper discusses various aspects of language modeling, such as normalization, its interaction with smoothing and class-based modeling.

2 Bayesian Classification and Language Modeling

The Bayesian classifier estimates the optimal class $\hat{c}$ given a sequence of words $w_1 \dots w_n$ where n is the length of the observation (i.e. a document to be classified) by the *prior* $P(c)$ and the *likelihood* $P(w_1 \cdots w_n|c)$:

$$\hat{c} = \arg\max_i P(w_1 \dots w_n|c_i) \cdot P(c_i) \tag{1}$$

We model the likelihood as a *Markov Chain*:

$$P(w_1 \dots w_n|c) = \prod_{i=1}^{n} P(w_i|w_{i-m+1} \dots w_{i-1}, c) \tag{2}$$

Each word w_i is considered with respect to some short history of preceding content $w_{i-m+1} \dots w_{i-1}$ where m is the size of the sequence to be modeled. We refer to this as an m-gram. The likelihood is estimated by different language models. Smoothing is essential in Bayesian classification since, otherwise, any unseen event would turn Equation 2 to zero. We experimented with the most common smoothing techniques in IR, as presented in [2], and discovered that *absolute discounting* works best in polarity classification which is why we used it in all subsequent experiments.

C. Macdonald et al. (Eds.): ECIR 2008, LNCS 4956, pp. 612–616, 2008.

3 Methods

3.1 Normalization

In text classification, one usually applies some form of normalization in order to reduce the data sparseness. Typically, one resorts to *stemming* – which is a simple algorithmic approach, where suffixes are removed from words – or to *lemmatization* – in which a base form is looked up in a dictionary. Due to its complexity, lemmatization is less preferred, though it is by far more linguistically accurate. In our experiments, we used Porter stemming and lemmatization as done in WordNet[1].

In order to remove noise, we also examined the effect of omitting *singletons* and *stopwords* being a small set of function words. Moreover, we restricted the vocabulary to the polarity expressions in *General Inquirer (GI)*[2] which is a list of polar adjectives, such as *brilliant* or *horrible*, and verbs, such as *adore* or *hate*, comprising 3440 different words in total. We assume that these polar words are highly discriminative for polar text classification.

In order to investigate the usefulness of negation handling in polar text classification, we examined the impact of two *negation* models which differ in complexity:

1. Each subsequent token of a negation marker, e.g. *not*, *didn't* or *cannot*, is marked with prefix *NOT_* until the first occurrence of a punctuation mark. This method has been proposed in [3].
2. With the help of regular expressions we disambiguate (potential) negation markers[3]. For this task we have written a small set of regular expressions. Unlike [3] only the negated word is marked with *NOT_*. We identify those words by part-of-speech information. Not only is this linguistically more accurate, but it should also cause less data-sparsity[4].

3.2 Class-Based Language Models

Unlike [1] who attempt to create more generalizing models by manually constructed rules replacing specific words with their respective part-of-speech tags, we try to generalize our training data by applying class-based language models. A mapping $k : V \rightarrow K$ is learned where V is the vocabulary and K is the set of unlabeled classes whose number has to be specified in advance. Unlike [1], this approach is completely *unsupervised* and does not require any form of expensive pre-processing, such as part-of-speech tagging. The objective function of the

[1] `http://wordnet.princeton.edu/`

[2] `http://www.wjh.harvard.edu/~inquirer`

[3] We observed that, frequently, negation markers do not express negations in certain contexts, e.g. ***not*** *just* ... *but* ... or *why* ***not*** ...

[4] Consider that each time a prefix *NOT* is added to a word, a new word is created which is different to the unnegated expression.

class-induction is the maximization of the *Likelihood*. The class-based language model is defined by:

$$P(w_i|w_{i-(m+1)} \dots w_{i-1}) = P(w_i|k(w_i)) \cdot P(k(w_i)|k(w_{i-(m+1)}) \dots k(w_{i-1})) \quad (3)$$

The first factor is called *emission probability* and the second is called *transition probability*. We use the $O(V \cdot K^2)$ algorithm as presented in [4] to learn the mapping from words to classes.

4 The Data

All our experiments were performed on the *movie review data* set [3]. We chose this dataset since it is commonly regarded as the benchmark dataset for polarity text classification. The dataset comprises 1000 positive and 1000 negative reviews. The classes to be predicted are *positive* and *negative* reviews. We randomly partitioned the dataset into a training set containing 936 documents, a development set for optimizing the language models and a test set both comprising 468 documents[5].

5 Results of the Experiments

We evaluated our experiments on the basis of *accuracy*. Every model has been optimally smoothed on a separate development set. We performed *four-fold cross-validation* meaning that the we generated four different partitions of the dataset as described in Section 4 in order to obtain representative numbers.

5.1 Results of Normalization Experiments

Table 1 displays the performance of the different types of normalizations. On the test set, only lemmatization and porter stemming perform marginally better than the plain unigram model. The remaining models, including both negation models, are worse than the plain unigram model though mostly only marginally. It is also striking that the two worst performing models, the GI model and the singleton model, are exactly those models from which the greatest amount of data has been removed. Apparently, it is fairly impossible to remove noise from the dataset we are using without also omitting meaningful information.

5.2 Interaction Between Smoothing and Normalization

Figure 1 illustrates the interaction between optimizing the smoothing parameter δ and the performance of some normalized models. If a suboptimal smoothing parameter has been chosen, e.g. $\delta = 0.1$, the GI model, which is the second

[5] These numbers have been chosen to be consistent with [3]. They use 1400 documents per class, using two thirds for training and one third for testing.

Table 1. Performance of different types of normalizations on unigram models

Model	Test	Devel
Plain	80.4	80.9
Porter Stemming	80.6	80.0
Lemmatization	**81.1**	81.0
Singletons Removed	64.6	64.0
Stopwords Removed	80.2	81.0
GI-Lexicon	78.7	80.6
Negation I	79.6	80.2
Negation II	79.7	81.0

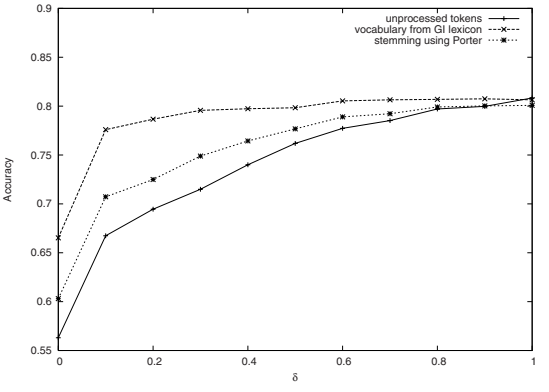

Fig. 1. Iterative optimization of smoothing parameters on unigram models on development set

worst performing model when optimized models are compared (see previous section), significantly outperforms the standard unigram model. Further iterating the smoothing parameter hardly improves the GI model but immensely improves the plain unigram model. The optimally smoothed plain model ($\delta = 0.95$) is, however, not only on a par with the optimal GI model on the development set but also better than the GI model on the test set. We observed a similar but less striking behavior between the model using stemming and the plain model. We conclude from these observations that expensive pre-processing, such as filtering the vocabulary with a manually built task-specific lexicon, does not offer a better performance than a properly smoothed plain unigram model, which is also far cheaper to obtain.

5.3 Results of Class-Based Language Models

In our experiments we found that trigram models work best as transition probabilities. We tested models with 500, 600 and 700 classes. Table 2 displays the results of the different class-based language models we built. For comparison,

Table 2. Performance of class-based language models

Model	Test	Devel
Unigram	80.4	80.9
Bigram	80.9	81.7
Trigram	81.3	81.5
500 Classes	81.7	82.2
600 Classes	**82.4**	81.7
700 Classes	82.3	82.8

we also included the performance of different word-based m-gram models. No normalization was done on any of the models. On the test set, all class-based models performed better than the word-based models, though the improvement is only limited. The class-based model trained on 600 classes with an accuracy of 82.4% is the best model we could generate in our experiments. Considering the error-bars on the results of our experiments at approximately $\pm 1.0\%$ this performance is comparable with SVMs at 82.9 which is the best performance reported in [3].

6 Conclusion

In this paper, we have stated our results on polar text classification using various types of language models. Properly smoothed plain unigram models offer similar performance to normalized models. Pre-processing is only beneficial if one uses insufficiently smoothed models. Removing noise often harms the performance. Class-based language models produce the best Bayesian classifier, though the gap to word-based higher-order m-gram models is small. Since our results are comparable to discriminative methods, such as optimized SVMs, we presume that the inherent noise in the data set does not allow much more room for improvement for Bayesian Classification and presumably any other standard machine learning algorithm.

References

1. Salvetti, F., Reichenbach, C., Lewis, S.: Impact of Lexical Filtering on Overall Opinion Polarity Identification. In: Proc. of the AAAI Symposium on Exploring Attitude and Affect in Text: Theories & Applications, Dordrecht, Netherlands (2004)
2. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In: Proc. of SIGIR, New Orleans, USA (2001)
3. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification Using Machine Learning Techniques. In: Proc. of EMNLP, Philadelphia, USA (2002)
4. Brown, P., Pietra, V.D., de Souza, P., Lai, J., Mercer, R.: Class-Based n-gram Models of Natural Language. Computational Linguistics 18(4), 467–479 (1992)