An Empirical Study of Skip-Gram Features and Regularization for Learning on Sentiment Analysis

Cheng Li(✉), Bingyu Wang, Virgil Pavlu, and Javed A. Aslam

College of Computer and Information Science, Northeastern University, Boston, MA, USA
{chengli,rainicy,vip,jaa}@ccs.neu.edu

Abstract. The problem of deciding the overall sentiment of a user review is usually treated as a text classification problem. The simplest machine learning setup for text classification uses a unigram bag-of-words feature representation of documents, and this has been shown to work well for a number of tasks such as spam detection and topic classification. However, the problem of sentiment analysis is more complex and not as easily captured with unigram (single-word) features. Bigram and trigram features capture certain local context and short distance negations—thus outperforming unigram bag-of-words features for sentiment analysis. But higher order n-gram features are often overly specific and sparse, so they increase model complexity and do not generalize well.

In this paper, we perform an empirical study of *skip-gram* features for large scale sentiment analysis. We demonstrate that skip-grams can be used to improve sentiment analysis performance in a model-efficient and scalable manner via regularized logistic regression. The feature sparsity problem associated with higher order n-grams can be alleviated by grouping similar n-grams into a single skip-gram: For example, "waste time" could match the n-gram variants "waste of time", "waste my time", "waste more time", "waste too much time", "waste a lot of time", and so on. To promote model-efficiency and prevent overfitting, we demonstrate the utility of logistic regression incorporating both L1 regularization (for feature selection) and L2 regularization (for weight distribution).

Keywords: Sentiment analysis · Skip-grams · Feature selection · Regularization

1 Introduction

The performance of sentiment analysis systems depends heavily on the underlying text representation quality. Unlike in traditional topical classification, simply applying standard machine learning algorithms such as Naive Bayes or SVM to *unigram* ("bag-of-words") features no longer provides satisfactory accuracy [19]. In sentiment analysis, unigrams cannot capture all relevant and informative features, resulting in information loss and suboptimal classification performance.

N. Ferro et al. (Eds.): ECIR 2016, LNCS 9626, pp. 72–87, 2016.
DOI: 10.1007/978-3-319-30671-1_6

For example, negation is a common linguistic construction that affects polarity but cannot be modeled by bag-of-words [24]. Finding a good feature representation for documents is central in sentiment analysis. Many rule and lexicon based methods are proposed to explicitly model negation relations [19,24]. However, rule and lexicon based approaches do not do well when the words' meanings change in specific domains. Researchers have found that applying machine learning algorithms to n-gram features captures some negations automatically and outperforms rule based systems [23]. For large datasets, frequent bigrams such as "not recommend" and "less entertaining" can model some negation-polarity word pairs. N-gram features are not only good at modeling short distance negations, but are also very useful in capturing subtle meanings, including implicit negations. For example, as mentioned in [19], the negative sentence "How could anyone sit through this movie?" contains no single negative unigram. However, the bigram "sit through" is a strong indicator for negative sentiment.

Although n-gram features are more powerful than unigrams, the diversity and variability of sentiment expressions sometimes makes strict n-gram matching hard to apply. Should the bigram "waste time" match the text "waste a lot of time"? A *skip-gram* [11] is an n-gram matched loosely in text, where looseness can be parameterized by a `slop` value, the number of additional words allowed in a matching span. For example with `slop=1` the skip-gram "waste time" would match "waste time", "waste of time", "waste more time", and "waste my time". With `slop=2` it would also match "waste of my time" and "waste too much time". With `slop=3` it can even match "waste a lot of time". The advantage of loose matching, informally, is that fewer features can match more phrases, which is good in several ways: First, it addresses the semantic matches that strict n-gram matching fails at, such as the n-gram "waste time" failing to match the text "waste my time". Second, higher order n-grams with $n > 3$ are often overly specific and sparse, and they only increase model complexity without generalizing well. Grouping similar n-grams to the same skip-gram alleviates this problem and makes learning more effective.

In this paper, the first research question we investigate is whether **skip-grams are good features** for large scale sentiment analysis when used by machine learning classifiers. We find that skip-gram features perform consistently better than unigram and n-gram features on all the data sets explored in our study. Skip-gram features also outperform word vector features on 2 of the 3 datasets we use, with inconclusive[1] results on the third (IMDB) dataset. We further investigate how varying the `slop` parameter affects sentiment analysis predictions. We generally find that `slop=1` helps as opposed to tight n-gram matches at `slop=0`, and increasing `slop` beyond 1 also helps, but to a lesser degree.

The second research question we investigate is an **appropriate learning mechanism that can handle such a large set of features**, addressing sparsity, speed, feature selection, and model-efficiency, all while retaining

[1] On the IMDB dataset, skip-grams perform worse than word vectors on the predefined test set, but better on randomly sampled test sets, as discussed in Sect. 3.

classification performance. An obvious concern when utilizing skip-grams with large size and `slop` is the large number of potential features generated. Even for the modest IMDB dataset with 50,000 reviews, we find that the number of potential skip-grams generated when using size up to 3 and `slop` up to 2 is nearly 2 million. For many learning algorithms, constructing a model from so many features is difficult. The model can overfit, and the prediction scores it produces are difficult to interpret. We investigate how to select a significantly smaller subset of features that yields good performance.

A natural way to deal with millions of features is to either employ feature selection *a priori* or to build selection into the training algorithm. The latter is often done via *regularization*. For example, L1-regularization [12] provides a convex surrogate to the L0 regularization, which linearly penalizes the number of features used in the model. Thus L1-regularization encourages a frugal use of features. In this paper, L1-regularization is our main mechanism for feature selection, demonstrated for Logistic Regression and SVM. In contrast, the other regularization often used—and one that we argue is necessary for sentiment analysis—is L2-regularization [12]. In the presence of correlated features, L2-regularization encourages the use of all correlates by explicitly penalizing large feature weights: small weights associated with multiple correlates will incur a lower penalty than a high weight associated with a single correlate. Having correlated features in the model can be very useful for generalizability and interpretability, at the cost of model efficiency as greater numbers of features are used. Hence, the L1- and L2-regularization trade-off: L1 encourages the frugal use of features, while L2 encourages judicious use of correlated features. For the problem of sentiment analysis, where sizable feature sets comprised of skip-grams generated with large n and `slop` are useful, we demonstrate the benefits of having a learning model that is both L1 and L2 regularized [12].

1.1 Related Work

Skip-Grams. For sentiment analysis on Twitter data, Fernández et al. [8] showed that using both n-grams and skip-grams give better performance than using n-grams alone. However, the dataset they used is small and tweets are usually short. It is unclear how many frequent skip-grams they extracted (the exact number is not mentioned). Also no feature selection is performed to pick informative skip-grams. As such, it is natural to ask whether skip-grams are still helpful on large datasets, where a huge number of frequent and possibly noisy skip-grams can be extracted. König and Brill [13] used skip-grams in deciding sentiments for movie review data and Microsoft customer feedback data via a multi-stage process. First, skip-gram candidates are generated based on a heuristic. Human assessors then review the skip-gram candidates and manually select the informative ones. At prediction time, a test document is checked to see if it matches any of the selected skip-grams; if it does, a label is assigned immediately based on the matched skip-gram; if not, a classifier trained on only n-grams is used to make a prediction. This hybrid approach is shown to work better than the standard method based purely on n-grams. However, it does not fully utilize

the power of skip-grams: since manual assessment is time consuming, only a very small number (300 in their experiments) of skip-gram candidates are generated and presented to the human assessors, and an even smaller number of features is kept. For a small number of selected skip-grams to work well, it is essential for them to be orthogonal so that different aspects of the data can be covered and explained. But skip-grams are judged independently of each other in both the automatic generating procedure and the human assessment procedure; as a result, individually informative features, when put together, could be highly correlated and redundant for prediction purposes. Also, the skip-grams selected are not used in conjunction with n-grams to train classifiers. In our proposed method, the feature selection is done by regularized learning algorithms, which is a much cheaper solution compared with manual selection. This reduction in cost makes it possible to generate and evaluate a large number of skip-gram candidates. The feature selection algorithm considers all features simultaneously, making the selected feature set less redundant.

Word Vectors. Another related line of research performs sentiment analysis based on word vectors (or paragraph vectors) [14,15,18]. Typical word vectors have only hundreds of dimensions, and thus represent documents more concisely than skip-grams do. One common way of building word vectors is to train them on top of skip-grams. After this training step, skip-grams are discarded and only word vectors are used to train the final classifier. Classifiers trained this way are smaller compared with those trained on skip-grams. One should note, however, that training classifiers on a low-dimensional dense word vector representation is not necessarily faster than training classifiers on a high-dimensional sparse skip-gram representation, for two reasons: first, low-dimensional dense features often work best with non-linear classifiers while high-dimensional sparse features often work best with linear classifiers, and linear classifiers are much faster to train. Second, sparsity in the feature matrix can be explored in the latter case to further speed up training. Although the idea of building word vector representations on top of skip-grams is very promising, current methods have some limitations. Documents with word vector representations are compressed or decoded in a highly complicated way, and the learned models based on word vectors are much more difficult to interpret than those based directly on skip-grams. For example, to understand what Amazon customers care about in baby products, it is hard to infer any latent meaning from a word vector feature. On the other hand, it is very easy to interpret a high-weight skip-gram feature such as "no smell", which includes potential variants like "no bad smell", "no medicine-like smell" and "no annoying smell". Another limitation is that, while word vectors are trained on skip-grams, they do not necessarily capture all the information in skip-grams. In our method, the classifiers are trained directly on skip-grams, and thus can fully utilize the information provided by skip-grams. We exploit the sparsity in the feature matrix to speed up training, and feature selection is employed to shrink the size of the classifier. Experiments show that our method generally achieves both better performance and better interpretability.

2 Learning with Skip-Gram Features

Extracting skip-grams from documents and computing matching scores is an IR, or NLP preprocessing problem which should be solved before the learning. We divide it into four steps: First, we lemmatize documents into tokens with the Stanford NLP package [1]. All stop-words are kept as they are often useful for sentiment analysis tasks. Second, preprocessed documents are sent to the search engine ElasticSearch [2] and an inverted index is built. Third, skip-gram candidates which meet the size, `slop` and document frequency requirements are gathered from the training document collection. To save memory and computation, skip-grams with very low document frequencies are discarded.

2.1 Skip-Gram Matching using ElasticSearch

In the last preprocessing step, we determine the matched documents for each of the skip-gram candidates and their matching scores. There are several slightly different ways of computing the matching score, but the basic idea is the same: a phrase that matches the given n-gram tightly contributes more to the score than a phrase that matches the n-gram loosely, and if two documents have the same skip-gram frequency, the shorter document will receive a higher score.

An indexing service is needed for storage and matching, i.e., a service such as Lemur [3], Terrier [4], Lucene [5] or ElasticSearch [2]. Any such platform can be used for this purpose. For this study, we adopt the **"Span Near Query"** scoring function implemented in the open source search engine ElasticSearch, which matches the above criteria. For a given (n-gram g, `slop` s, document d) triple,

$$\mathbf{score}(g, s, d) = \sqrt{\frac{\mathbf{skipGramFreq}(g, s, d)}{\mathbf{length}(d)}}$$

where

$$\mathbf{skipGramFreq}(g, s, d) = \sum_{k=0}^{s} \frac{\mathbf{phraseFreq}(g, k, d)}{\mathbf{length}(g) + 1 + k}$$

and $\mathbf{phraseFreq}(g, k, d)$ is the number of phrases in d generated by inserting k extra words in the given n-gram. We further normalize $\mathbf{score}(g, s, d)$ to the range [0,1].

2.2 Learning Algorithms and Regularization

After matching each skip-gram against the document collection, we obtain a feature matrix which can be fed into most classification algorithms. We use regularized SVM with a linear kernel and regularized Logistic Regression (LR). Both are linear models, thus fast to train and less likely to overfit high dimensional data.

Regularized SVM minimizes thesum of hinge loss and a penalty term [7]. Specifically, for L2-regularized SVM, the objective is

$$\min_w \sum_{i=1}^{N} (\max(0, 1 - y_i w^T x_i))^2 + \lambda \frac{1}{2} ||w||_2^2,$$

and for L1-regularized SVM, the objective is

$$\min_w \sum_{i=1}^{N} (\max(0, 1 - y_i w^T x_i))^2 + \lambda ||w||_1,$$

where λ controls the strength of the regularization[2]. We use the LibLinear package [7] for regularized SVM. Using both L1 and L2 terms to regularize SVM has been proposed [22], but is not commonly seen in practice, possibly due to the difficult learning procedure; hence we do not consider it in this study.

Regularized LR [12] minimizes the sum of logistic loss and some penalty term. Specifically, for L2-regularized LR, the objective is

$$\min_w -\frac{1}{N} \sum_{i=1}^{N} y_i w^T x_i + \log(1 + e^{w^T x_i}) + \lambda \frac{1}{2} ||w||_2^2,$$

for L1-regularized LR, the objective is

$$\min_w -\frac{1}{N} \sum_{i=1}^{N} y_i w^T x_i + \log(1 + e^{w^T x_i}) + \lambda ||w||_1,$$

and for L1+L2-regularized LR, the objective is

$$\min_w -\frac{1}{N} \sum_{i=1}^{N} y_i w^T x_i + \log(1 + e^{w^T x_i}) + \lambda \alpha ||w||_1 + \lambda (1 - \alpha) \frac{1}{2} ||w||_2^2.$$

In L1+L2 LR, the L1 ratio α controls the balance between L1 regularization and L2 regularization. When $\alpha = 0$, the model has only the L2 penalty term; when $\alpha = 1$, the model has only the L1 penalty term.

Unlike the case for SVM, LR with both L1 and L2 penalties is widely adopted, possibly due to the efficient training and hyper-parameter tuning algorithms available [10]. However, we find that the most popular package glmnet [9] for L1+L2 LR does not scale well on our datasets which contain hundreds of thousands of documents with millions of skip-gram features. Thus we make use of our own Java implementation[3], which has a special optimization for sparse matrix representations and is more scalable than glmnet.

3 Experiments

Datasets and setup. To examine the effectiveness of skip-grams, we extract skip-gram features from three large sentiment analysis datasets and train several

[2] In the LibLinear package that we use, a different notation is used; there $C = 1/\lambda$.

[3] Our code is publicly available at https://github.com/cheng-li/pyramid.

machine learning algorithms on these extracted features. The datasets used are IMDB, Amazon Baby, and Amazon Phone. IMDB [15] contains 50,000 labeled movie reviews. Reviews with ratings from 1 to 4 are considered negative; and 7 to 10 are considered positive. Reviews with neutral ratings are ignored. The overall label distribution is well balanced (25,000 positive and 25,000 negative). IMDB comes with a predefined train/test split, which we adopt in our experiments. There are also another 50,000 unlabeled reviews available for unsupervised training or semi-supervised training, which we do not use. Amazon Baby (containing Amazon baby product reviews) and Amazon Phone (containing cell phone and accessory reviews) are both subsets of a larger Amazon review collection [17]. Here we use them for binary sentiment analysis the same manner as in IMDB dataset. By convention, reviews with rating 1–2 are considered negative and 4–5 are positive. The neutral ones are ignored. Amazon Baby contains 136,461 positive and 32,950 negative reviews. Amazon Phone contains 47,970 positive and 22,241 negative reviews. Amazon Baby and Amazon Phone do not have a predefined train/test partitioning. We perform stratified sampling to choose a random 20 % of the data as the test set. All results reported below on these two datasets are averaged across five runs.

For each dataset, we extract skip-gram features with max size n varying from 1 (unigram) to 5 (5-gram) and max `slop` varying from 0 (no extra words can be added) to 2 (maximal 2 words can be added). For example, when max n=2 and max `slop`=1, we will consider unigrams, bigrams, and skip-bigrams with `slop`=1. As a result, for each dataset, 13 different feature sets are created. The combinations (max n=1, max `slop`=1) and (max n=1, max `slop`=2) are essentially the same as (max n=1, max `slop`=0), and thus not considered. For each feature set, we run five learning algorithms on it and measure the accuracies on the test set. The algorithms considered are L1 SVM, L2 SVM, L1 LR, L2 LR and L1+L2 LR. In order to make the feature set the only varying factor, we use fixed hyper parameters for all algorithms across all feature sets. The hyper parameters are chosen by cross-validation on training sets with unigram features. For L2 SVM, $C = 1/\lambda = 0.0625$; for L1 SVM, $C = 1/\lambda = 0.25$; for LR, $\lambda = 0.00001$; and for L1+L2 LR, $\alpha = 0.1$. Performing all experiments took about five days using a cluster with six 2.80 GHz Xeon CPUs.

3.1 Main Results

Figure 1 shows how increasing max n and max `slop` of the skip-grams affects the logistic regression performance on Amazon Baby. In each sub-figure, the bottom line is the performance with standard n-gram (max `slop`=0) features. Along each bottom line, moving from unigrams (max n=1) to bigrams (max n=2) gives substantial improvement. Bigrams such as "not recommend" are effective at capturing short distance negations, which cannot be captured by unigrams.

Moving beyond bigrams (max n=2) to higher order n-grams, we can see some further improvement, but not as big as before. This observation is consistent with the common practice in sentiment analysis, where trigrams are not commonly

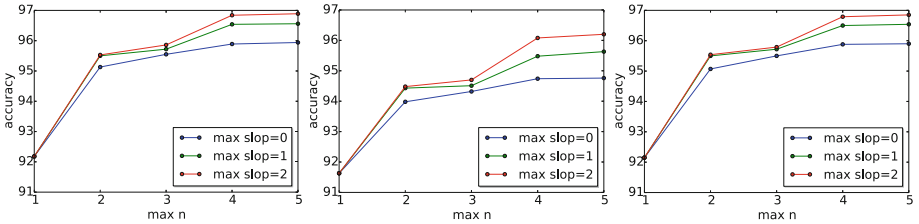

Fig. 1. Performance of LR on Amazon Baby with skip-gram features of varying sizes and `slop`s. Left to right learning algorithms: L2-LR, L1-LR, L1+L2-LR.

Table 1. The performance of our method on Amazon Baby. For each algorithm on each feature set, the table shows its test accuracy and the number and fraction of features selected. The accuracies which are significantly better (at 0.05 level under t-test) than those by a corresponding `slop` 0 baseline are marked with *.

max n, `slop`	L2 SVM	L2 LR	Features used	L1 SVM	Features used	L1 LR	Features used	L1+L2 LR	Features used
1, 0	92.38	92.18	8×10^3 (100 %)	92.36	4×10^3 (49 %)	91.63	2×10^3 (23 %)	92.15	7×10^3 (88 %)
2, 0	95.12	95.13	6×10^4 (100 %)	94.89	8×10^3 (12 %)	93.98	3×10^3 (5 %)	95.07	3×10^4 (58 %)
3, 0	95.51	95.55	1×10^5 (100 %)	95.20	9×10^3 (6 %)	94.32	4×10^3 (2 %)	95.50	6×10^4 (45 %)
4, 0	95.88	95.89	1×10^5 (100 %)	95.59	9×10^3 (5 %)	94.74	4×10^3 (2 %)	95.88	7×10^4 (41 %)
5, 0	95.93	95.94	1×10^5 (100 %)	95.59	9×10^3 (5 %)	94.76	4×10^3 (2 %)	95.90	7×10^4 (40 %)
2, 1	95.51*	95.50*	1×10^5 (100 %)	95.22*	1×10^4 (6 %)	94.43*	4×10^3 (2 %)	95.50*	8×10^4 (45 %)
3, 1	95.70*	95.72*	5×10^5 (100 %)	95.36*	1×10^4 (2 %)	94.51*	5×10^3 (1 %)	95.72*	1×10^5 (25 %)
4, 1	96.51*	96.54*	6×10^5 (100 %)	96.19*	1×10^4 (1 %)	95.48*	5×10^3 (<1 %)	96.50*	1×10^5 (20 %)
5, 1	96.56*	96.56*	6×10^5 (100 %)	96.23*	1×10^4 (1 %)	95.63*	5×10^3 (<1 %)	96.54*	1×10^5 (19 %)
2, 2	95.51*	95.53*	3×10^5 (100 %)	95.19*	1×10^4 (3 %)	94.48*	6×10^3 (1 %)	95.54*	1×10^5 (32 %)
3, 2	95.89*	95.86*	1×10^6 (100 %)	95.41*	1×10^4 (1 %)	94.70*	6×10^3 (<1 %)	95.79*	1×10^5 (15 %)
4, 2	96.85*	96.84*	1×10^6 (100 %)	96.57*	1×10^4 (<1 %)	96.08*	6×10^3 (<1 %)	96.79*	1×10^5 (10 %)
5, 2	96.87*	96.89*	1×10^6 (100 %)	96.60*	1×10^4 (<1 %)	96.20*	7×10^3 (<1 %)	96.85*	1×10^5 (9 %)

used compared with bigrams, and n-grams beyond trigrams are rarely used. When we fix the max n and increase the max `slop`, we see the performance further improves. For $n \geq 4$, increasing max `slop` often brings more improvement than increasing max n. Similar observations can be made for SVM and for the other two datasets.

Table 2. The performance of our method on IMDB. For each algorithm on each feature set, the table shows its test accuracy and the number and fraction of features selected.

max n, `slop`	L2 SVM	L2 LR	Features used	L1 SVM	Features used	L1 LR	Features used	L1+L2 LR	Features used
1, 0	89.10	88.58	2×10^4 (100 %)	88.81	2×10^3 (9 %)	88.59	3×10^3 (12 %)	88.71	1×10^4 (75 %)
2, 0	90.81	90.63	1×10^5 (100 %)	90.02	2×10^3 (1 %)	89.80	3×10^3 (2 %)	90.62	6×10^4 (40 %)
3, 0	91.10	91.02	2×10^5 (100 %)	90.13	2×10^3 (<1 %)	89.84	3×10^3 (1 %)	90.89	8×10^4 (28 %)
4, 0	91.19	91.13	3×10^5 (100 %)	90.19	2×10^3 (<1 %)	89.85	3×10^3 (<1 %)	90.97	9×10^4 (25 %)
5, 0	91.21	91.16	4×10^5 (100 %)	90.18	2×10^3 (<1 %)	89.85	3×10^3 (<1 %)	90.96	9×10^4 (24 %)
2, 1	91.22	91.13	3×10^5 (100 %)	90.24	3×10^3 (<1 %)	90.01	3×10^3 (<1 %)	90.94	9×10^4 (25 %)
3, 1	91.46	91.44	9×10^5 (100 %)	90.26	3×10^3 (<1 %)	90.07	3×10^3 (<1 %)	91.20	1×10^5 (13 %)
4, 1	91.56	91.54	1×10^6 (100 %)	90.37	3×10^3 (<1 %)	90.11	3×10^3 (<1 %)	91.22	1×10^5 (11 %)
5, 1	91.65	91.64	1×10^6 (100 %)	90.36	3×10^3 (<1 %)	90.15	3×10^3 (<1 %)	91.24	1×10^5 (10 %)
2, 2	91.32	91.35	6×10^5 (100 %)	90.37	2×10^3 (<1 %)	90.07	4×10^3 (<1 %)	90.96	1×10^5 (17 %)
3, 2	91.65	91.60	2×10^6 (100 %)	90.40	3×10^3 (<1 %)	90.23	4×10^3 (<1 %)	91.25	1×10^5 (7 %)
4, 2	91.76	91.64	2×10^6 (100 %)	90.43	3×10^3 (<1 %)	90.26	4×10^3 (<1 %)	91.23	1×10^5 (5 %)
5, 2	91.71	91.63	3×10^6 (100 %)	90.43	3×10^3 (<1 %)	90.26	4×10^3 (<1 %)	91.26	1×10^5 (5 %)

Tables 1, 2, and 3 show more detailed results on these datasets. For each fixed n, we use a paired t-test (0.05 level) to check whether increasing max `slop` from 0 to 1 or 2 leads to significant improvement. On Amazon Baby, all improvements due to the increase of max `slop` are significant. On Amazon Phone, about half are significant. The significance test is not done on the IMDB dataset since only the predefined test set is used.

Tables 1, 2, and 3 also show how many features are selected by each learning algorithm. L2 regularized algorithms do best in terms of accuracy but at the cost of using all features. If that is acceptable in certain use cases, then L2 regularization is recommended. On the other hand, L1 regularization can greatly reduce the number of features used to below 1 %, sacrificing test accuracy by 1–2 %; if this drop in performance is acceptable, then L1 regularization is recommended for the extremely compact feature sets produced. Finally L1+L2 regularization is a good middle choice for reducing the number of features to about 5–20 % while at the same time maintaining test accuracy on par with L2 regularization.

Table 3. The performance of our method on Amazon Phone. For each algorithm on each feature set, the table shows its test accuracy and the number and fraction of features selected. The accuracies which are significantly better (at 0.05 level under t-test) than those by a corresponding `slop` 0 baseline are marked with *.

max n, `slop`	L2 SVM	L2 LR	Features used	L1 SVM	Features used	L1 LR	Features used	L1+L2 LR	Features used
1, 0	89.45	89.27	5×10^3 (100 %)	89.36	2×10^3 (45 %)	89.33	2×10^3 (41 %)	89.30	5×10^3 (96 %)
2, 0	92.03	91.89	3×10^4 (100 %)	91.82	5×10^3 (14 %)	91.85	4×10^3 (13 %)	91.94	2×10^4 (79 %)
3, 0	92.24	92.11	6×10^4 (100 %)	91.77	5×10^3 (8 %)	91.92	5×10^3 (8 %)	92.14	4×10^4 (70 %)
4, 0	92.44	92.26	8×10^4 (100 %)	91.94	5×10^3 (6 %)	91.99	5×10^3 (6 %)	92.32	5×10^4 (66 %)
5, 0	92.29	92.19	8×10^4 (100 %)	91.89	5×10^3 (6 %)	92.01	5×10^3 (6 %)	92.28	5×10^4 (65 %)
2, 1	92.39	92.25	9×10^4 (100 %)	92.19	6×10^3 (6 %)	92.30*	6×10^3 (7 %)	92.29	6×10^4 (65 %)
3, 1	92.31	92.33	2×10^5 (100 %)	92.09*	7×10^3 (3 %)	92.27	8×10^3 (3 %)	92.33*	9×10^4 (43 %)
4, 1	92.31	92.43	2×10^5 (100 %)	92.18	7×10^3 (2 %)	92.28	8×10^3 (2 %)	92.42	1×10^5 (38 %)
5, 1	92.33	92.37	3×10^5 (100 %)	92.15	7×10^3 (2 %)	92.31*	8×10^3 (2 %)	92.31	1×10^5 (36 %)
2, 2	92.57*	92.67*	4×10^5 (100 %)	92.13*	8×10^3 (1 %)	92.32*	1×10^4 (2 %)	92.53*	1×10^5 (28 %)
3, 2	92.53	92.64*	4×10^5 (100 %)	92.13	8×10^3 (1 %)	92.37*	1×10^4 (2 %)	92.55*	1×10^5 (28 %)
4, 2	92.59	92.74*	6×10^5 (100 %)	92.13	8×10^3 (1 %)	92.30	1×10^4 (1 %)	92.58*	1×10^5 (23 %)
5, 2	92.54*	92.67*	7×10^5 (100 %)	92.24*	8×10^3 (1 %)	92.40*	1×10^4 (1 %)	92.58*	1×10^5 (22 %)

3.2 Comparisons with Other Methods

For the IMDB dataset, public results on the predefined test set are listed in Table 4. Among the methods which only use labeled data, our method based on skip-grams achieved the highest accuracy. Paragraph vectors (based on word2vec) trained on both labeled data and unlabeled data achieve noticeably higher performance. In fact, using one public paragraph vector implementation[4], with only labeled data and a RBF SVM classifier[5], we are able to produce 93.56 % accuracy on the given test set. However, the performance of paragraph

[4] The paragraph vector implementation is from https://github.com/klb3713/sentence2vec/. The parameters we use are size=400, alpha=0.025, window=10, min_count=5, sample=0, seed=1, min_alpha=0.0001, sg=1, hs=1, negative=0, cbow_mean=0.

[5] After producing paragraph vectors, we run LIBSVM (https://www.csie.ntu.edu.tw/~cjlin/libsvm/) with c=32, g=0.0078. An RBF kernel performs better than a linear kernel.

Table 4. Our approach compared to other methods on the IMDB dataset.

Classifier	Features	Training documents	Accuracy
LR with dropout regularization [21]	bigrams	25,000 labeled	91.31
NBSVM [23]	bigrams	25,000 labeled	91.22
SVM with L2 regularization	structural parse tree features + unigrams [16]	25,000 labeled	82.8
LR L1+L2 regularization	5-grams selected by compressive feature learning [20]	25,000 labeled	90.4
SVM	word vectors trained by WRRBM [6]	25,000 labeled	89.23
SVM	word vectors [15]	25,000 labeled + 50,000 unlabeled	88.89
LR with dropout regularization [21]	bigrams	25,000 labeled + 50,000 unlabeled	91.98
LR	paragraph vectors [14]	25,000 labeled + 50,000 unlabeled	92.58
LR with L2 regularization	skip-grams	25,000 labeled	91.63
SVM with L2 regularization	skip-grams	25,000 labeled	91.71
LR with L1+L2 regularization	skip-grams	25,000 labeled	91.26

vectors seems quite sensitive to the specific training/testing partitioning. After re-partitioning the data randomly (50 %–50 % as before), the accuracy of paragraph vectors based on the same hyper-parameters dropped significantly to only around 85 %. By contrast, our method consistently produces high results on both the given test set and randomly sampled test set.

Amazon review datasets are often used differently by different researchers, which makes the published results not directly comparable. Here we train paragraph vectors on the same subset of documents and report the performance.[6] On Amazon Baby, paragraph vector gives 88.84 % while our method gives 96.85 %. On Amazon Phone, paragraph vector gives 85.38 % while our method gives 92.58 %.

4 Analysis of Skip-Grams

When designing a feature set, the primary concern is often generalizability, since good generalizability implies good prediction performance. In sentiment analysis

[6] The training parameters are the same as in IMDB.

data, people often express the same idea in many slightly different ways, which makes the prediction task harder as the algorithm has to learn many expressions with small variations. Skip-grams alleviate this problem by letting the algorithm focus on the important terms in the phrase and tolerate small changes in unimportant terms. Thus skip-grams perform feature grouping on top of n-grams without requiring any external domain knowledge. This not only improves generalizability but also interpretability. Several such skip-gram examples are shown in Table 5. They are selected by an L1+L2 regularized logistic regression model with high weights. For each skip-gram, we show its count in the entire collection and several n-gram instances that it matches. For each matched n-gram, the count in the collection is also listed in the table. We can see, for example, the skip-gram "only problem" (`slop`=1) could match bigram "only problem" and trigrams "only minor problem" and "only tiny problem". Although the bigram "only problem" is frequent enough in the collection, the trigram "only tiny problem" only occurs in four out of 169,411 reviews. It is hard for the algorithm to treat the trigram "only tiny problem" confidently as a positive sentiment indicator. After grouping all such n-gram variants into the same skip-gram, the algorithm can assign a large positive weight to the skip-gram as a whole, thus also handling the rare cases properly. This also provides more concise rules and facilitates user interpretation.

4.1 Feature Utility

We analyze to what extent skip-gram features contribute to overall performance. Take the Amazon Phone dataset as an example. The skip-gram features in it can be broken down into different types based on n and `slop` values. The left column of Fig. 2 shows, when max $n = 3$ and max `slop`=2, about 85 % of the extracted skip-gram features have non-zero `slop`s. In the middle column in Fig. 2, we only focus on features selected by L1+L2 logistic regression and recheck their count distribution. The fraction of unigrams increases, while the fraction of `slop`=2 trigrams decreases. One can imagine that many noisy/irrelevant `slop`=2 skip-trigrams are eliminated by the L1 regularization, and unigrams are less noisy. We further sum the logistic regression weights (absolute values, which are comparable since all features are normalized) for features within each type and display the results in the right column. The standard n-grams with `slop=0` only contribute to 20 % of the total weight, and the remaining 80 % is due to skip-grams with non-zero `slop`s.

4.2 Feature Selection for Skip-Grams

Grouping similar n-grams into skip-grams not only produces generalizable features but sometimes also noisy features. For example, in Table 5, "I have to return", "I have never had to return", "I finally have to return" and "I do not have to return" are all grouped into the skip-gram "I have to return" (`slop`=2).

Table 5. Examples of high weight skip-grams for LR.

Skipgram and count		Matched ngrams and count			
skip movie (slop 2)	42	skip this movie	28	skip this pointless movie	1
		skip the movie	8	skipping all the movies	1
		skip watching this movie	1	of this sort	
it fail (slop 1)	358	it fails	279	it completely fails	5
		it even fails	5	it simply fails	3
whole thing (slop 1)	729	whole thing	682	whole horrific thing	1
		whole damn thing	5		
waste time (slop 1)	1562	waste time	109	waste of time	676
		waste your time	4	waste more time	6
only problem (slop 1)	1481	only problem	1378	only tiny problem	4
		only minor problem	11		
never leak (slop 2)	1053	never leak	545	never a urine leak problem	1
		never have leak	86	never have any leak	77
no smell (slop 1)	445	no smell	340	no medicine-like smell	1
		no bad smell	13	no annoying smell	5
it easy to clean and (slop 2)	314	it is easy to wipe clean and	3	it is easy to keep clean and	3
		it is so easy to clean and	16		
I have to return (slop 2)	216	I have to return	151	I finally have to return	1
		I have never had to return	1	I do not have to return	4
good service (slop 2)	209	good service	131	good price and service	1
		good and fast service	2		

This is the worst kind of noise because the gap matches negation words and different instances of the skip-gram have opposite sentiments. Detecting and modeling the scope of negations is very challenging in general [24]. We do not deal with negations at skip-gram generation time; at learning time, we rely on feature selection to eliminate such noisy skip-grams. In this particular example, the noise is relatively low as the mismatched n-grams "I have never had to return" and "I do not have to return" are very rare in the document collection. Therefore logistic regression still assigns a large weight to this skip-gram. Some other skip-grams are more likely to include negations and are thus more noisy. For example, the skip-gram "I recommend" (`slop`=2) can match many occurrences of both "I highly recommend" and "I do not recommend". Our feature selection mechanism infers that this skip-gram does more harm than good and assigns a small weight to it. In practice, we find the denoising effect of feature selection to be satisfactory. Most of the classification mistakes are not caused by skip-gram mismatch but due to the inability to identify the subjects of the sentiment expressions: many reviews compare several movies/products and thus the algorithm gets confused as to which subject the sentiment expression should apply. Resolving this issue requires other NLP techniques and is beyond the scope of this study.

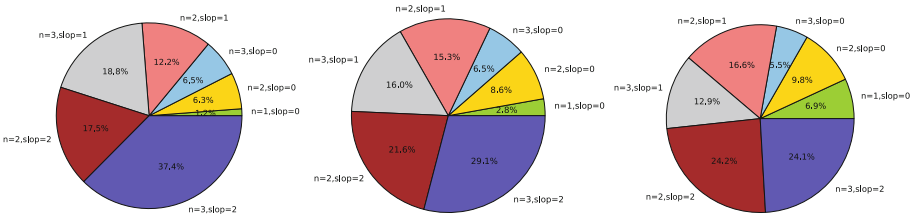

Fig. 2. L1+L2 LR selected features for Amazon Phone feature contribution analysis, max n =3. LEFT: feature count distribution in dataset; MIDDLE: feature count distribution of selected features; RIGHT: feature LR-weighted-distribution of selected features

From Tables 1, 2, and 3, it is very clear that L2 regularization achieves better overall accuracy than L1 regularization. This seems counter-intuitive because L1 regularization completely eliminates noisy features while L2 regularization only shrinks their weights. We believe there are two main reasons for this: First, the document collections are relatively big. The bigger the dataset is, the more parameters can be reliably estimated. L1 regularization is very successful at "large p, small n" problems where the sample size is often in the hundreds while the feature space could be in the millions. Our sentiment analysis datasets, however, are much larger, and this fact makes it possible for L2 logistic regression to estimate almost all parameters. In this case, assigning very low (not necessarily exactly 0) weights to noisy features will suffice. Second, in the presence of many highly correlated features, L1 regularization usually picks only one of them and discards the rest. But the same opinion/sentiment is often expressed in many different ways, which means L1 regularization's instability in handling correlated features can hurt the prediction performance.

But performance is not the only factor we care about. Having an L1 regularization can produce smaller models, which makes the prediction faster and the model more interpretable. L1+L2 regularization provides a good balance between model compactness and prediction accuracy, since a relatively small fraction of features is selected and the performance does not appreciably suffer. In all three datasets, if we limit the number of features used to be under 1×10^5, then the best performance is always achieved by L1+L2 LR, trained on skip-grams of maximum size 5 and slop 2.

5 Conclusion

We demonstrate that *skip-grams* can be used to improve large scale sentiment analysis performance in a model-efficient and scalable manner via regularized logistic regression. We show that although n-grams beyond trigrams are often very specific and sparse, many similar n-grams can be grouped into a single skip-gram which benefits both model-efficiency and classification performance.

To promote model-efficiency and prevent overfitting, we demonstrate the utility of logistic regression incorporating both L1 regularization (for feature selection) and L2 regularization (for weight distribution). L2 regularized algorithms do best in terms of accuracy but at the cost of using all features. L1 regularization can greatly reduce the number of features used to below 1 %, sacrificing test accuracy by 1–2 %. L1+L2 regularization is a good middle choice for reducing the number of features significantly while maintaining good test accuracy.

Acknowledgments. The research is supported by NSF grant IIS-1421399.

References

1. http://nlp.stanford.edu/software/
2. https://lucene.apache.org/
3. http://www.lemurproject.org/
4. http://terrier.org/
5. http://www.elasticsearch.org/
6. Dahl, G.E., Adams, R.P., Larochelle, H.: Training restricted Boltzmann machines on word observations. arXiv preprint (2012). arxiv:1202.5695
7. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: Liblinear: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
8. Fernández, J., Gutiérrez, Y., Gómez, J.M., Martınez-Barco, P.: Gplsi: supervised sentiment analysis in twitter using skipgrams. In: SemEval 2014, pp. 294–299 (2014)
9. Friedman, J., Hastie, T., Tibshirani, R.: glmnet: Lasso and elastic-net regularized generalized linear models. R package version, 1 (2009)
10. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. **33**(1), 1 (2010)
11. Guthrie, D., Allison, B., Liu, W., Guthrie, L., Wilks, Y.: A closer look at skip-gram modelling. In: LREC-2006, pp. 1–4 (2006)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, vol. 2. Springer, New York (2009)
13. König, A.C., Brill, E.: Reducing the human overhead in text categorization. In: KDD, pp. 598–603. ACM (2006)
14. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. arXiv preprint (2014). arxiv:1405.4053
15. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: ACL 2011, pp. 142–150. Association for Computational Linguistics (2011)
16. Massung, S., Zhai, C., Hockenmaier, J.: Structural parse tree features for text representation. In: ICSC, pp. 9–16. IEEE (2013)
17. McAuley, J., Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 165–172. ACM (2013)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint (2013). arxiv:1301.3781

19. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10, pp. 79–86. Association for Computational Linguistics (2002)
20. Paskov, H.S., West, R., Mitchell, J.C., Hastie, T.: Compressive feature learning. In: NIPS, pp. 2931–2939 (2013)
21. Wager, S., Wang, S., Liang, P.S.: Dropout training as adaptive regularization. In: NIPS, pp. 351–359 (2013)
22. Wang, L., Zhu, J., Zou, H.: The doubly regularized support vector machine. Statistica Sinica **16**(2), 589 (2006)
23. Wang, S.I., Manning, C.D.: Baselines and bigrams: simple, good sentiment and topic classification. In: Proceedings of the ACL, pp. 90–94 (2012)
24. Wiegand, M., Balahur, A., Roth, B., Klakow, D., Montoyo, A.: A survey on the role of negation in sentiment analysis. In: Proceedings of the Workshop on Negation and Speculation in Natural Language Processing, pp. 60–68. Association for Computational Linguistics (2010)