Product Title Classification versus Text Classification

Hsiang-Fu Yu* Chia-Hua Ho† Prakash Arunachalam‡
Manas Somaiya‡ Chih-Jen Lin†

Abstract

In most e-commerce platforms, product title classification is a crucial task. It can assist sellers listing an item in an appropriate category. At first glance, product title classification is merely an instance of text classification problems, which are well-studied in literature. However, product titles possess some properties very different from general documents. A title is usually a very short description, and an incomplete sentence. A product title classifier may need to be designed differently from a text classifier, although this issue has not been thoroughly studied. In this work, using a large-scale real-world data set, we examine conventional text-classification procedures on product title data. These procedures include word stemming, stop-word removal, feature representation and multi-class classification. Our major findings include that stemming and stop-word removal are harmful, and bigrams or degree-2 polynomial mappings are very effective. Further, if linear classifiers such as SVMs are applied, instance normalization does not downgrade the performance and binary/TF-IDF representations perform similarly. These results lead to a concrete guideline for practitioners on product title classification.

1 Introduction

With many e-commerce platforms nowadays, product title classification becomes crucial to assist sellers listing an item in an appropriate category. Although sellers may provide additional descriptions, some systems perform classification right after obtaining titles because of using class-dependent forms subsequently. At first sight, by considering a product title as a short text, title classification is just a variant of the well-developed area of text

*Department of Computer Science, The University of Texas at Austin. Email: rofuyu@cs.utexas.edu

†Department of Computer Science, National Taiwan University. Email: {b95082,cjlin}@csie.ntu.edu.tw

‡eBay Inc. Email: {parunachalam,msomaiya}@ebay.com

classification. While many mature techniques have been proposed in each phase of text classification from the initial preprocessing to the final categorization, they may not be suitable for product titles. Titles are different from texts in several aspects such as the length of each instance (most titles are very short), the distribution of lengths (most product titles have similar lengths), and the grammatical structure (most titles are incomplete sentences).

Existing systems for text classification, such as WEKA [1] and NLTK [2], often consider a bag-of-word approach so that each feature corresponds to one or several words. The procedure begins with a preprocessing phase to unify words in various morphological forms (stemming) and retain useful lexicons (stop-word removal), followed by a conversion from texts to feature vectors (e.g., TF-IDF model). Then classification methods (e.g., linear SVM and Naive Bayes) are applied on a labeled corpus to obtain a classifier for categorizing new texts. If the performance is not satisfactory, more features such as bigrams or trigrams may be added to provide additional information. Currently, practitioners working on product title classification face the situation of whether to apply these conventional steps or not. We have witnessed that engineers painfully try many combinations. Our goal in this paper is to identify key differences between product title classification and general text classification. Then a guideline can be available for practitioners.

Product title classification can be applied in various scenarios. Because our aim is to identify general properties, we limit our scope by considering the following settings.

1. We try not to touch application or data dependent issues. For example, some data sets are very noisy and require data cleaning. We skip such issues because they may not be general for all product title data.

2. We consider the situation of not having too many classes. In some places, product titles are categorized into a complicated taxonomy of thousands of classes. We focus on classifying level-1 meta-categories of the taxonomy, so the number of classes is less than 100. This simplified setting is still useful in practice because many hierarchical classification methods handle the large number of classes by first considering a coarse-level classification task (e.g., [3,4] for hierarchical text classification and [5] for hierarchical title classification).

Our main contributions include

1. We identify properties of product title classification different from those of text classification. For example, stemming and stop-word removal are not needed, and bigrams or polynomial mappings are very effective. A list of our findings are in Section 9.

2. We demonstrate that because titles are short, the analysis of title classification can be conducted on both "micro" (details of a wrongly predicted title) and "macro" (the performance of a whole set) levels.

3. We extensively study how linear SVM is applied to large-scale multi-class title classification.

Some works (e.g., [5]) have targeted at product title classification, although ours differs from them by aiming at identifying general properties of product titles. A related but broader area is short-text classification, which already has rich literature. Examples include question classification (e.g., [6, 7]) and sentence classification (e.g., [8]). However, these data types differ in some aspects; for example, titles are noun phrases but questions are complete sentences. Throughout our discussion, we will mention results in these existing works. For job title classification, [9] proposes a useful approach different from the traditional bag-of-word model, but their titles are shorter than ours. Recently for short-text classification, many works (e.g., [10, 11]) enrich the features by using additional resources. This type of approaches is beyond our scope here, because we solely consider title texts. Product title classification is also related to the topic "product categorization" (e.g., [12, 13]), which often assumes more information such as price.

The paper is organized as follows. We discuss conventional procedures for general text classification in Section 2, followed by Section 3 describing the data set and the baseline approach we use. By analyzing errors made by the baseline, we design experiments in subsequent sections. Section 4 focuses on the effect of two preprocessing techniques, stemming and stop-word removal. In Section 5, we examine various feature representations. Different multi-class strategies are investigated in Section 6. In Section 7, we further examine the effect of bigram and polynomial-mapping features. Other issues are discussed in Section 8. Finally, we give conclusions and future issues in Section 9.

2 Conventional Procedures of Text Classification

We introduce the following commonly used techniques for text classification.

2.1 Stemming and Stop-word Removal

To improve the performance and reduce the number of features, two commonly used pre-processing methods are stemming and stop-word removal. Stemming unifies some different words with the same stem. For example, in English, "check," "checked," and "checking" have the same stem "check." Many algorithms have been designed for stemming, e.g., Lovins Stemmer [14] and Porter Stemmer [15].

It is known that some words with very high frequencies are not very helpful for text classification. For example, prepositions or pronouns like "this" and "of" may not be very informative. Removing these "stop words" can possibly improve the performance of text classification.

Stemming/stop-word removal have been shown to be useful (e.g., [16]) and not useful (e.g., [8]) in different scenarios. See more discussion in, for example, [17].

2.2 Feature Representation

In a bag-of-word representation, how to give each word a feature value is an issue. The simplest method is a binary representation so that each feature is 0 or 1 to indicate the appearance of a word. This setting can be easily extended to term frequency (TF), which indicates the number of occurrences of a word in a text. Some more sophisticated feature representations have been proposed. TF-IDF [18] is a popular one defined as

$$x_{i,j} = \mathrm{TF}_{i,j} \times \mathrm{IDF}_j, \tag{1}$$

where x_{ij} is the jth feature of the ith feature vector $\boldsymbol{x}_i$, $\mathrm{TF}_{i,j}$ is the term frequency, and IDF_j is the inverse document frequency,

$$\mathrm{IDF}_j = \log\left(\frac{\#\text{training instances}}{\#\text{training instances with word } j}\right).$$

Because popular words may appear in the same document several times and cause large TF values, IDF is used to avoid their dominance.

A common step after generating feature vectors is to normalize each instance to a unit vector. That is,

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i / \|\boldsymbol{x}_i\|.$$

2.3 Multi-class Classification

Many classification methods have been applied to text classification. Because it is not possible to study many methods here, we focus on linear support vector machines (SVM) [19], which have been shown to be among the best methods for text classification [20]. Besides, recent advances in large-scale linear classification have created efficient implementations for huge text document sets; see a survey in [21].

Given training feature vectors $\boldsymbol{x}_i \in \mathcal{R}^n$ and labels $y_i \in \{-1, 1\}$, $i = 1, \ldots, l$, SVM solves the following problem.

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l} \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i), \tag{2}$$

where $C > 0$ is the regularization parameter. The second term in Eq. (2) indicates the sum of training losses. Given a new test instance $\boldsymbol{x}$, the predicted label is $\text{sgn}(\boldsymbol{w}^T\boldsymbol{x})$.

To extend two-class SVM to solve multi-class problems, the most used approach is the one-versus-rest method [22]. For data in classes $\{1, \dots, k\}$, this method obtains k models, $\boldsymbol{w}^1, \dots, \boldsymbol{w}^k$, each of which is from training one class as positive and others as negative. Following the design in Eq. (2), for an instance $\boldsymbol{x}$ with label y, ideally we should have

$$(\boldsymbol{w}^y)^T\boldsymbol{x} \geq 1 \quad \text{and} \quad (\boldsymbol{w}^m)^T\boldsymbol{x} \leq -1, \ \forall m \neq y. \tag{3}$$

Therefore, the predicted label $\bar{y}$ is

$$\arg \max_{m=1,\dots,k} (\boldsymbol{w}^m)^T\boldsymbol{x}.$$

We will discuss two other multi-class methods in Section 6.

2.4 Unigram, Bigram, and Other Features

In a bag-of-word model, if each feature corresponds to a word, then we have unigram features. However, splitting a phrase into separate words may lose some information. For example, "checking" and "account" may not be useful to indicate the concept of "checking account." In this case, the bigram "checking account" should be considered as a feature. In text classification and natural language processing, bigrams, n-grams (n consecutive words), or item sets (n words co-occurred in a document) have been commonly added as features (e.g., [23]). However, such features may not be always useful, as indicated in the survey of bigrams [24] and the work of item-set features [25]. We will deeply study bigram features in Section 7.

3 Data, Baseline and Error Analysis

In this section, we begin with preparing some real-world data sets for experiments. Then we devise a baseline setting. By analyzing errors made by the baseline, we design subsequent experiments in this paper to investigate the difference between product title classification and text classification.

3.1 Data

Our data set comes from a large Internet company. The training set is obtained by the following steps.

1. We collect all sold items in a three-month period.
2. We remove few categories. Either their business is independently run or they are very close to some others. In the end, we consider 29 classes of data.

Table 1: Data statistics. The number of classes is 29.

	#instances	#features	#non-zero feature values
Training	10,000,001	603,840	104,536,500
Test	7,310,307	880,418	75,855,772

3. We subsample 10M instances by a stratified setting to keep the class distribution.

For the test set, in a period after the three-month one for obtaining training data, we collect about about 7M sold items.

The setting of using sold items is similar to [5] because labels of such items are considered more accurate. An issue is whether we should also include unsold items for training. We will discuss this issue in Section 8.1.

The statistics of training and test data are in Table 1. The reason why we have 10M + 1 training instances is because one class contains very few instances. Our code for stratified subsampling, after selecting 10M instances, ensures that at least one instance per class is selected. Most other classes are reasonably balanced, contributing about 1% to 10% of the training set. Regarding the number of features, in Table 1, the 0.6M features correspond to unique tokens appeared in the training set. The test set contains 0.28M new tokens, so our feature ID is up to 0.88M (0.6M + 0.28M). In fact, our training and test sets share only 0.33M tokens. From Table 1, we also notice that on average each training/test instance includes around 10 tokens.

We run all experiments on a machine with Intel Xeon 3.07GHz CPU (X5675), 12MB Cache, and 74GB RAM.

3.2 Baseline

We consider the following simple setting as the baseline.

- Conversion of all words into lower case.
- Simple tokenization by splitting titles on spaces, punctuations and number/alphabet transitions. For example, "aa:bb 70d" becomes four tokens "aa," "bb," "70," and "d."
- No stemming; no stop-word removal.
- Binary unigram as feature values.
- One-versus-rest multi-class SVM for classification.

For SVM, we use a popular linear classification package LIBLINEAR [26]. The parameter C in (2) is selected by five-fold cross validation (CV) on values $C \in \{2^{-4}, 2^{-3}, \ldots, 2^{0}\}$. We briefly discuss how to decide the search space of C. It is known [27] that for linear SVM, there is a value C^* such that

$$\text{optimal solution } \boldsymbol{w} \text{ of Eq. (2) is the same after } C \geq C^*.$$

Therefore, it is not needed to keep increasing C if CV rates have stabilized. The longer training time of using a large C is another concern. See more discussion in Section 5.

For evaluation, because of the request of the data provider, we are not able to present the absolute error rates. Instead, we report the relative error rate to the baseline.

$$\text{RE}_{\text{baseline}} \equiv \frac{\text{error}(\bar{\boldsymbol{y}})}{\text{error}(\bar{\boldsymbol{y}}_{\text{baseline}})} \times 100\%,$$

where $\bar{\boldsymbol{y}}$ is the result for evaluation and $\bar{\boldsymbol{y}}_{\text{baseline}}$ is the prediction by the baseline on the test set. Therefore, the baseline's $\text{RE}_{\text{baseline}}$ is 100% and we hope to develop approaches having as small $\text{RE}_{\text{baseline}}$ as possible.

3.3 Error Analysis

The result of running the baseline is reasonably good. An investigation shows there are three types of test errors.

1. The instance is wrongly labeled. This situation is common in real-world applications. For example, in our data set, an item "*southwest colors sunrise cascade necklace & earring set*" is considered as a "sporting goods," but classifying it in the "jewelry & watches" class is more appropriate. We can design some methods to correct wrongly labeled data, although this topic is beyond the scope of this paper. Here we simply accept their existence and do not attempt to correct their labels.

2. The instance can be considered in more than one class. For example, "*sport watch*" can be put either in "jewelry & watches" or "sporting goods." To handle this type of data, we need a multi-label rather than a multi-class method. This topic is beyond the scope of the paper, so we do not particularly handle such data here.

3. The instance is wrongly classified. For example, "*new deadfish link fishing earrings set dead bone fish*" is clearly a "sporting goods," but wrongly classified as in "jewelry & watches." This type of instances is what we are interested in. We hope to know why they are wrongly classified under a typical text classification setting and what possible remedies are.

We identify some wrongly classified data and check the corresponding weight values in the model vectors. Two examples are shown in Table 2 with the following information.

- Tokens of the title.
- w_j^y, where y is the true class and j is the index of a token.
- $w_j^{\bar{y}}$, where $\bar{y}$ is the predicted class.
- Number of token j's appearances in the whole training set, in class y, and in class $\bar{y}$.

Table 2: Analysis of wrongly predicted instances by the baseline.

(a) tokenized title: "*rubber d ring 5 bit*"

$(\boldsymbol{w}^y)^T\boldsymbol{x} = -0.891101,\ (\boldsymbol{w}^{\bar{y}})^T\boldsymbol{x} = 0.582337$

Tokens	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	Number of appearances		
			training	class y	class $\bar{y}$
d	-0.1391	-0.0929	219,129	6,298	10,320
5	-0.0072	-0.1018	662,933	48,549	91,888
ring	-0.5869	1.6495	147,793	2,407	129,080
rubber	-0.2889	-0.0419	44,021	2,588	5,386
bit	0.1309	-0.8306	9,397	1,136	120

(b) tokenized title: "*3 stainless steel split rings pack of 25*"

$(\boldsymbol{w}^y)^T\boldsymbol{x} = -0.37592,\ (\boldsymbol{w}^{\bar{y}})^T\boldsymbol{x} = -0.279519$

Tokens	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	Number of appearances		
			training	class y	class $\bar{y}$
3	-0.0686	-0.1421	980,520	47,991	59,113
25	-0.0310	-0.1570	103,606	6,498	13,070
of	-0.1786	-0.0589	504,914	14,511	28,665
pack	0.0763	-0.7400	99,808	9,235	551
rings	-0.1308	1.3139	27,069	2,563	17,285
stainless	-0.2404	0.2306	46,674	3,595	26,855
steel	0.0366	0.1712	69,034	10,804	31,839
split	0.1605	-0.8973	4,745	950	540

We have the following observations from Table 2.

1. Some words dominate the prediction of the instance. For example, in Table 2(a), the word "ring" causes the title to be classified as a "jewelry & watches" item. We have

$$w_{\text{ring}}^{\text{jewelry \& watches}} \gg 0 \text{ but } w_{\text{ring}}^{\text{sporting goods}} < 0.$$

   Based on the fact that 129,080 of 147,793 training instances containing the token "ring" are in the "jewelry & watches" class, the one-versus-rest classifier correctly identifies that "ring" is an important word for this class.

2. In Table 2(b), we have that the number of appearances of "pack" in the class "sporting goods" is 16.7x more than that in the class "jewelry & watches." Thus, we would hope that "pack" is a useful word to distinguish these two classes. However, the number of appearances in the class "sporting goods" is not large enough to make "pack" an important feature in training "sporting goods" versus "non-sporting goods." We suspect that because the one-versus-rest method independently considers k two-class problems, for any two classes y_1 and y_2, and a feature j, the relation between $w_j^{y_1}$ and $w_j^{y_2}$ may not be well adjusted. It will be interesting to check if other multi-class methods perform better or not.

3. In Table 2(a), we see neither "5" nor "bit" can be useful to distinguish "sporting goods" and "jewelry & watches." However, a further check shows that "5" and "bit" only co-occur in "sporting goods" rather than "jewelry & watches." Using a bigram feature such as "5 bit" may be useful in this situation.

We call the investigation in Table 2 a micro-level analysis because details of an instance are presented. This is in contrast to the setting of checking the global performance, which we call a macro-level analysis. Note that we can easily conduct micro-level analyses because titles are short.

In subsequent experiments, we will check observations made from the above error analysis.

4 Effect of Stemming and Stop-word Removal

We consider Porter Stemming [15] and a stop-word list provided in SenseClusters.[1] In Table 3, we present results after applying stemming or/and stop-word removal. By "Both" in the table, we apply stemming on both data and the stop-word list first, and then perform stop-word removal. From Table 3, clearly, both stemming and stop-word removal deteriorate the performance.

[1] `http://www.d.umn.edu/~tpederse/senseclusters.html`

Table 3: A comparison of stemming and removing stop words. $\text{RE}_{\text{baseline}}$: lower is better.

	$\text{RE}_{\text{baseline}}$
Baseline	100.00%
Stop-word Removal	102.53%
Stemming	105.93%
Both	108.91%

This result might not be too surprising because we have mentioned in Section 2.1 that stemming/stop-word removal are not always helpful. For short-text classification, although earlier works such as [5] and [8] have suspected or observed that stemming/stop-word removal are harmful, Table 3 may be the first to quantitatively show that the difference is significant.

To gain more insights, in Table 4, we give some examples that are originally correctly predicted by the baseline, but become wrong after stemming and stop-word removal. Similar to Table 2, we present $\boldsymbol{w}^y$ and $\boldsymbol{w}^{\bar{y}}$, where y and $\bar{y}$ are true and predicted classes, respectively. The example in Table 4(a) is in the class "consumer electronics." After stemming, the word "recorder" becomes "record," so it is then predicted to the class "music." For the example in Table 4(b), it is correctly predicted to "home & garden" by the baseline because in the training set there are similar titles such as "*barbie all dolled up lunch dinner plates 8 new.*" After stemming, the word "dolled" becomes "doll." Because $w_{\text{doll}}^{\text{dolls \& bears}} \gg 0$, the instance becomes wrongly predicted.

Examples in Table 4 show that stemming may remove some words special for a class. Because product titles are short, this operation seems to be harmful rather than beneficial. Further, various stemming approaches are available; they range from grouping only some words to grouping many [16]. The approach we take is a more conservative one. Therefore, we suspect that if a more aggressive stemming method is applied, the performance will be even worse.

However, applying stemming and removing stop words do not always mean worse results. In some examples not shown here, after stemming, a more appropriate weight is assigned. This observation shows that for any modeling technique, it can be helpful in some aspects, but harmful in some others. The decision relies on its overall impact. In Table 5, we illustrate this point by showing numbers of correctly/wrongly predicted instances by the baseline and the approach of applying stemming and stop-word removal. Following the same reason explained earlier, we can not present numbers here. Instead, we let η be the number of instances wrongly predicted by the baseline, but correctly predicted after stemming/stop-word removal. We then present other values as the ratios to η. From Table 5, applying stemming/stop-word removal improves η instances, but ruins

Table 4: Sample test instances correctly predicted by baseline but wrongly predicted after stemming/stop-word removal.

(a) tokenized title: "*sony cd radio cassette recorder*"

true(y): "consumer electronics" predicted($\bar{y}$): "music"

baseline	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	both	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$
cd	-0.2998	2.4577	cd	-0.3980	2.4974
radio	0.9950	-0.0059	radio	1.1012	-0.0327
sony	0.0185	-0.8003	soni	-0.0065	-0.9148
recorder	0.7471	-2.5978	record	0.3280	1.0750
cassette	0.2412	1.1197	cassett	0.2900	1.0628

(b) tokenized title: "*barbie all dolled up birthday party jointed banner*"

true(y): "home & garden" predicted($\bar{y}$): "dolls & bears"

baseline	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	both	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$
all	-0.0976	-0.1373	doll	-1.8254	2.2961
up	-0.1247	-0.1807	parti	0.8412	-0.3423
party	0.7907	-0.3652	barbi	-2.1698	3.0364
barbie	-2.1829	2.9152	birthdai	0.8397	-0.5943
birthday	0.8413	-0.5609	joint	0.1312	0.8592
banner	0.1339	-1.3000	banner	0.1876	-1.2433
jointed	-0.0750	0.9290			
dolled	1.0000	-2.0000			

Table 5: The numbers of wrongly predicted items. η is the number of items which are wrongly predicted by the baseline but correctly predicted after stemming and stop-word removal.

		Stemming/stop-word removal	
		correct	wrong
Baseline	correct	—	1.77η
	wrong	η	7.65η

Table 6: A comparison of various feature-representation methods. $\text{RE}_{\text{baseline}}$: lower is better.

	$\text{RE}_{\text{baseline}}$	Training (s)	Test (s)
Baseline (binary)	100.00%	1,550.98	20.77
Binary + unit-length	99.93%	1,012.29	20.92
TF-IDF + unit-length	99.74%	1,449.30	21.40

1.77η instances. Therefore, overall stemming and stop-word removal are not beneficial.

5 Effect of Various Feature Representations

Following the discussion in Section 2.2, we compare three feature representations.

1. Baseline: each feature is binary (i.e., 0 or 1) to indicate the occurrence of a word.

2. Binary + unit-length: it is the same as baseline, but each instance is normalized to have unit length.

3. TF-IDF: see Eq. (1), but we use binary (0/1) values as TF. Each instance is normalized to have unit length.

A crucial difference from general text classification is that in a product title of around 10 words, usually words are unique. This is why for TF-IDF we simply use 0/1 values as TF. In [6] for question classification, the authors have pointed out this property and used binary rather than TF representations.

For the second and the third representations, because data have been normalized, we slightly shift the search space of the SVM penalty parameter C to $\{2^{-2}, 2^{-1}, \ldots, 2^{2}\}$. See explanation later in this section.

In Table 6, we present both relative error rates to the baseline and training/test time. Clearly, the error rates of the three representations are similar. However, the training time of binary representation after normalization is

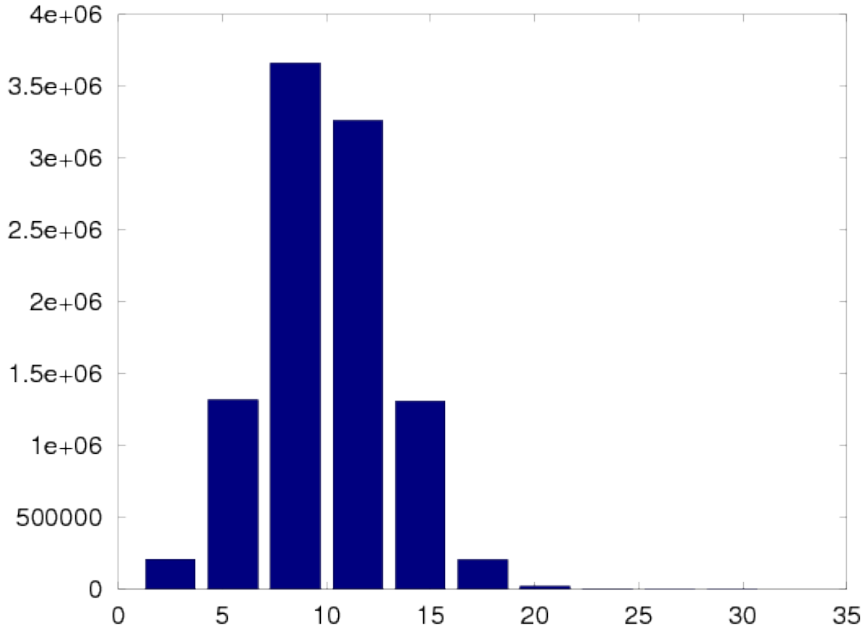

Figure 1: The histogram of the number of tokens in training instances.

shorter. We give the following explanation. It is well known that for certain SVM solvers including what we have used here, a larger penalty parameter C causes longer running time.[2] A simple derivation in the Appendix shows that

$$\begin{aligned} & \text{SVM on data } \boldsymbol{x}_i, \forall i \text{ with the parameter } C \\ \equiv\ & \text{SVM on data } \Delta\boldsymbol{x}_i, \forall i \text{ with the parameter } C/(\Delta^2), \end{aligned} \tag{4}$$

where Δ is any non-zero constant. For example, if $\|\boldsymbol{x}_i\| = 10, \forall i$, then baseline with $C = 1$ is equivalent to training normalized data with $C = 100$. Therefore, if data is not normalized, easily the parameter C tried is too large and causes lengthy training time. We have observed in our experiments that if the search space of C includes large values, then the training time rapidly increases. The relationship in Eq. (4) also explains why for normalized data, we shift the search space of C to include larger values.

The above analysis implies that if $\|\boldsymbol{x}_i\|, \forall i$ are similar, then binary representation with and without normalization gives similar performance (assuming suitable parameter selection such as CV has been conducted). This property generally holds for product titles because of similar lengths. In Figure 1, we draw the histogram of the number of tokens in training instances. Clearly, for most instances, the number of tokens is between 4 and

[2]See, for example, Section 5 of the practical guide in [26]'s appendix.

12. In contrast, for text classification, the length of articles may significantly vary, so normalization reduces the training time under the risk of obtaining different test accuracy. Based on the discussion, normalization is always applied in subsequent experiments.

We observe that TF-IDF gives a similar error rate. The reason is likely because TF values are usually 0 or 1 (indeed we directly use 0/1 values as explained earlier). In this situation, TF-IDF essentially feature-wisely scales training instances so the following problem is solved.

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l} \max(0, 1 - y_i \boldsymbol{w}^T D\boldsymbol{x}_i),$$

where D is a diagonal matrix. If the regularization term $\|\boldsymbol{w}\|^2/2$ is not considered and D is invertible,[3] then

$$\min_{\boldsymbol{w}} \quad C\sum_{i=1}^{l} \max(0, 1 - y_i \boldsymbol{w}^T D\boldsymbol{x}_i) \tag{5}$$

$$\equiv \min_{\bar{\boldsymbol{w}}} \quad C\sum_{i=1}^{l} \max(0, 1 - y_i \bar{\boldsymbol{w}}^T \boldsymbol{x}_i), \tag{6}$$

where Eq. (5) and Eq. (6) correspond to optimization problems of using TF-IDF and binary, respectively. Their optimal solutions $\boldsymbol{w}^*$ and $\bar{\boldsymbol{w}}^*$ satisfy $\bar{\boldsymbol{w}}^* = D\boldsymbol{w}^*$. For any new $\boldsymbol{x}$, its TF-IDF representation is $D\boldsymbol{x}$, so

$$(\boldsymbol{w}^*)^T D\boldsymbol{x} = (\bar{\boldsymbol{w}}^*)^T \boldsymbol{x}$$

implies the same decision value. This derivation shows that because TF of product titles is generally 0 or 1, if linear classifiers such as SVM or logistic regression are applied, the performance of the TF-IDF representation should be similar to that of the binary representation.

6 Effect of Different Multi-class Classifiers

In Section 3.3, from some experimental results, we suspect that in some situations one-versus-rest may not obtain appropriate weights because k binary models are independently trained. Therefore, we experiment with two other common multi-class linear SVM methods.

1. One-versus-one multi-class method [28]. For k classes of data, this method generates $k(k-1)/2$ models, each of which involves only two classes of training data. For testing, although there are many approaches, voting is a simple and commonly used strategy. If the model of class i versus class j predicts i, then the counter for class i is increased by one. In the end, the class with the highest number of votes is predicted. The main

[3] D is invertible if $\text{IDF}_j > 0$ for all j. A title set can rarely have $\text{IDF}_j = 0$ because it means word j appears in all instances.

difficulty of applying the one-versus-one approach here is that it needs $O(k^2n)$ storage to store $O(k^2)$ models. Because of this reason, in the recent survey for large-scale linear classification [21], the authors conclude that the one-versus-one approach is less practical. Further, LIBLINEAR does not support this multi-class method. We create an implementation by storing $k(k-1)/2$ models in a sparse form. Some technical details will be discussed in Section 8.3.

2. Multi-class SVM by solving one single optimization problem. Existing approaches of this type include [29–31], but here we mainly consider the approach by Crammer and Singer [30]. It solves the following problem.

$$\begin{aligned}\min_{\boldsymbol{w}^1,\dots,\boldsymbol{w}^k} \quad & \frac{1}{2}\sum_{m=1}^{k} \|\boldsymbol{w}^m\|^2 + \\ & C\sum_{i=1}^{l} \max_{m:m\neq y_i} \max(0, 1-(\boldsymbol{w}^{y_i}-\boldsymbol{w}^m)^T\boldsymbol{x}_i).\end{aligned}$$

The loss term $\max_{m:m\neq y_i}(\cdots)$ indicates that ideally we have

$$(\boldsymbol{w}^{y_i})^T\boldsymbol{x}_i - (\boldsymbol{w}^m)^T\boldsymbol{x}_i \geq 1,\ \forall m \neq y_i.$$

This concept is very similar to Eq. (3) for one-versus-rest. However, the main difference is that all k models are obtained together from one optimization problem.

In addition to the performance, we are interested in training/test time. Table 7 gives the comparison results. The error rates of the three methods are similar, although one-versus-one and Crammer and Singer are slightly better. Unfortunately, we find that it is harder to analyze the three methods here. In Sections 4 and 5, we solve the same optimization problem, but train different data. In contrast, we now solve different optimization problems using the same data. Further, we have inconsistent results because later in Section 7, with more features in the data, one-versus-rest becomes competitive. After some preliminary analysis, we cannot conclude which method is better in terms of error rates and decide to leave this issue for future investigation.

For training time, Crammer and Singer SVM is clearly the fastest. This finding is interesting. In the paper [32] introducing LIBLINEAR's implementation for Crammer and Singer's SVM, the authors compare with one-versus-rest on test accuracy only. LIBLINEAR applies coordinate-descent methods for all three multi-class approaches compared here. We particularly check one-versus-rest and Crammer and Singer because their concepts are very similar. We find that the timing difference is not because of different numbers of operations. Instead, memory access is the reason. For one-versus-rest, the number of training-data accesses is

$$k \times l \times \#\text{iterations}, \tag{7}$$

Table 7: A comparison of multi-class SVM methods. $\text{RE}_{\text{baseline}}$: lower is better.

	$\text{RE}_{\text{baseline}}$	Training (s)	Test (s)
One-versus-rest	99.93%	1,012.29	20.92
One-versus-one	95.28%	732.38	82.63
Crammer & Singer	97.30%	175.77	20.82

where #iterations is the average number of iterations taken by the k two-class problems and each iteration is a cycle of l coordinate descent steps to go through all training instances. In contrast, for Crammer and Singer, the number is only

$$l \times \#\text{iterations}. \tag{8}$$

The reason is that each time when an instance is accessed, it is used for simultaneously updating k models. In our experiment, on average each two-class training of one-versus-rest needs 90 iterations, while Crammer and Singer's SVM needs 60. However, the running time of one-versus-rest is five times rather 1.5 times more than that of Crammer and Singer. The reason is because of the different numbers of data accesses shown in Eqs. (7) and (8).

For test time, because of $k(k-1)/2$ inner products, one-versus-one is slower than the other two even though we have sped up its test using a sparse storage of the model.

The conclusion of this section is that because of similar performance but faster training, Crammer and Singer SVM is preferred.

7 Effect of Bigram and Polynomial Mappings

We conjectured in Section 3.3 that for product title classification some pairs of words may provide useful information. In Table 8, we compare the following two methods to generate more features.

1. Bigram + unigram.
2. Degree-2 polynomial mapping (poly2). A unigram feature vector $\boldsymbol{x} \in \mathcal{R}^n$ is expanded to

$$[1, x_1, \ldots, x_n, x_1^2, \ldots, x_n^2, x_1x_2, \ldots, x_{n-1}x_n]. \tag{9}$$

The method poly2 is related to item-set features (n words co-occurred in a document) used in text classification (e.g., [25] and [33]). The number of features in Eq. (9) can be up to $n(n+1)/2$, which is 10^{11} in our case. Fortunately, some features never have non-zero values in the training set. To efficiently remove unnecessary features, we implement a Hashing technique

Table 8: A comparison of different methods to generate more features. $\text{RE}_{\text{baseline}}$: lower is better.

Bigram + Unigram (#Features = 11,799,345)

	$\text{RE}_{\text{baseline}}$	Training (s)	Test (s)
One-versus-rest	71.54%	1,559.37	74.43
One-versus-one	75.26%	1,090.76	148.11
Crammer & Singer	75.14%	615.34	73.93

Poly2 (#Features = 41,689,205)

	$\text{RE}_{\text{baseline}}$	Training (s)	Test (s)
One-versus-rest	70.13%	3,983.46	280.17
One-versus-one	74.13%	2,812.21	535.19
Crammer & Singer	71.21%	1,531.79	271.38

proposed in Section 5 of [34]. The numbers of features, shown in Table 8, are rather large: 11M for bigrams and 41M for poly2. We normalize each instance to have unit length. The search space for the SVM parameter C is $\{2^{-2}, 2^{-1}, \ldots, 2^2\}$. Because the three multi-class strategies perform similarly in Section 6, we run all of them here.

Table 8 shows that these two types of features give much better results than those in Table 7 by unigrams. We have also checked the error rate of each class and found that results are consistently better (details not shown here). Further, for all three multi-class strategies, poly2 is slightly better than bigram + unigram. For the best result, the number of errors is reduced to only 70% of the baseline.

To investigate the usefulness of our newly added features, we analyze the same two titles used in Section 3.3. By the same setting to show $\boldsymbol{w}^{\text{sporting goods}}$ and $\boldsymbol{w}^{\text{jewelry \& watches}}$, and the number of word occurrences in the training set, we present results in Table 9. Both titles were wrongly predicted to "jewelry & watches," but with bigram features, they are now correctly predicted as in "sporting goods." From Table 9(a), "d ring" and "5 bit" occur more often in "sporting goods," so they help to remedy the issue caused by "ring," which mainly appears in the "jewelry & watches" class. Similarly, in Table 9(b), "split rings" and "steel split" contribute to the correct prediction of the instance.

We then identify some titles which are wrongly predicted by bigram + unigram, but correctly predicted by poly2. In an example shown in Table 10, among all the token combinations, "sydney olympics" is a useful feature for the prediction to the correct class "sports mem, cards & fan shop." However, this feature is not a bigram because "sydney" and "olympics" are separated in the title "sydney 2000 olympics ..." Therefore, the classifier of using bigrams wrongly predicts the instance to the "dvds & movies" class because of the high weights of "opera" and "house" in $\boldsymbol{w}^{\text{dvds \& movies}}$. This

Table 9: Predicting the two titles in Table 2 by bigrams.

(a) tokenized title: "*rubber d ring 5 bit*"

y: "sporting goods" $\bar{y}$: "jewelry & watches"

$(\boldsymbol{w}^y)^T\boldsymbol{x} = 2.87354,\ (\boldsymbol{w}^{\bar{y}})^T\boldsymbol{x} = 0.827066$

Tokens	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	Number of appearances		
			training	class y	class $\bar{y}$
d	-0.0514	-0.0522	219,129	6,298	10,320
5	0.2466	0.0368	662,933	48,549	91,888
ring	0.0757	2.8543	147,793	2,407	129,080
rubber	0.3394	0.2492	44,021	2,588	5,386
bit	1.3889	-0.7510	9,397	1,136	120
ring 5	-0.1316	0.1958	580	6	538
d ring	0.5881	-1.3797	244	63	16
5 bit	0.4179	-0.3260	22	11	0
rubber d	0.0000	0.0000	7	0	0

(b) tokenized title: "*3 stainless steel split rings pack of 25*"

y: "sporting goods" $\bar{y}$: "jewelry & watches"

$(\boldsymbol{w}^y)^T\boldsymbol{x} = 5.92099,\ (\boldsymbol{w}^{\bar{y}})^T\boldsymbol{x} = 1.47078$

Tokens	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	Number of appearances		
			training	class y	class $\bar{y}$
3	0.1066	-0.0185	980,520	47,991	59,113
25	0.1004	0.0207	103,606	6,498	13,070
of	-0.1525	0.0157	504,914	14,511	28,665
pack	0.5651	-0.3614	99,808	9,235	551
rings	0.5100	2.4874	27,069	2,563	17,285
pack of	0.1497	-0.0455	5,891	535	101
stainless	0.9438	1.0372	46,674	3,595	26,855
steel	0.5115	0.4972	69,034	10,804	31,839
stainless steel	-0.3909	-0.0373	37,551	2,483	23,468
split	0.2342	-0.0098	4,745	950	540
split rings	1.4936	-0.5728	403	211	175
of 25	0.2679	0.0262	2,369	150	274
steel split	1.9943	-1.4904	67	65	2
rings pack	-0.1293	-0.0000	13	0	0
3 stainless	-0.2833	-0.0781	66	8	16

Table 10: An example which is wrongly predicted by bigram but correctly predicted by poly2.

tokenized title: "*sydney 2000 olympics parthenon opera house*"
true(y): "sports mem, cards & fan shop" predicted($\bar{y}$): "dvds & movies"

Degree-2 Polynomial Mapping

Features	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	Number of appearances		
			training	class y	class $\bar{y}$
constant "1" in Eq. (9)	-0.63	0.53	10,000,001	523,911	240,493
sydney	0.33	0.20	707	58	31
2000	0.57	0.01	41,511	5,064	3,033
olympics	1.21	0.17	2,113	463	26
parthenon	-0.22	-0.00	37	2	1
opera	-0.65	0.20	2,255	7	242
house	-0.11	0.36	21,993	339	1,628
sydney sydney	0.33	0.20	707	58	31
2000 2000	0.57	0.01	41,511	5,064	3,033
olympics olympics	1.21	0.17	2,113	463	26
parthenon parthenon	-0.22	-0.00	37	2	1
opera opera	-0.65	0.20	2,255	7	242
house house	-0.11	0.36	21,993	339	1,628
sydney 2000	0.19	0.23	95	43	2
sydney olympics	0.42	0.00	32	12	0
sydney opera	0.10	-0.01	27	3	3
sydney house	0.10	0.00	25	3	3
olympics 2000	0.40	0.00	39	14	0
2000 opera	0.10	0.00	6	3	0
2000 house	0.09	-0.08	65	6	12
olympics opera	0.09	0.00	1	1	0
olympics house	0.06	-0.02	8	1	0
opera house	0.02	-0.05	77	3	12

Unigram + Bigram

Features	$\boldsymbol{w}^y$	$\boldsymbol{w}^{\bar{y}}$	Number of appearances		
			training	class y	class $\bar{y}$
house	-0.17	0.54	21,993	339	1,628
opera	-1.01	0.39	2,255	7	242
2000	0.61	0.14	41,511	5,064	3,033
olympics	1.89	0.45	2,113	463	26
sydney	0.69	0.51	707	58	31
parthenon	-0.55	0.13	37	2	1
sydney 2000	0.33	0.52	46	24	1
opera house	0.34	-0.12	76	3	12
2000 olympics	0.23	0.00	10	3	0

example shows that "long distance relationship" of words may be helpful. We can use and present such information only because titles are short.

The improvement made by bigrams or poly2 in Table 8 seems to be more significant than what reported earlier for text classification. In Table 1 of [21], for five text sets, the accuracy of using linear SVM is almost the same as nonlinear SVM using a Gaussian kernel. Past studies [34] have indicated that the performance of using degree-2 polynomial mappings tends to be somewhere in between. Therefore, although we have not conducted experiments, for the five document sets used in [21], neither bigram nor degree-2 polynomial mappings may significantly improve the performance. The average document length of these five sets is at least 50. Therefore, because tokens of a longer document may have given rich enough information, word pairs do not provide additional information. In fact, in an earlier survey on bigrams [24], the authors conclude that "for an unrestricted text categorization task one would probably not expect dramatic effects of using bigrams." In contrast, because of short title lengths, bigram or degree-2 polynomial features are very useful for title classification.

However, we must point out that in [6, 7] for question classification, the authors show no improvements using n-grams. Because both questions and titles are short texts, we may expect a similar conclusion in their and our works. However, data and many settings are different, so more detailed investigations are needed.

8 Other Issues

We discuss some issues related to our design of experiments and our implementations.

8.1 Training/test Data Preparation

In Section 3.1, we generated data using sold items because their class labels are considered more reliable. However, unsold items may provide additional information (e.g., some rare words) for training. We prepare another training set of 10M instances by subsampling from all listed items rather than sold items. We then train a model to predict the same test set of sold items used in earlier experiments. The test accuracy is slightly lower. This result shows that if the evaluation is on sold items, using also sold items for training is better because of closer data distributions.

8.2 More Data or More Features

In data classification, a possible way to improve the performance is by collecting more training data. In this paper, up to now we consider 10M training instances subsampled from all sold items, and investigate various mod-

Table 11: A comparison: more data versus more features. $\text{RE}_{\text{baseline}}$: lower is better.

	#instnaces	#features	$\text{RE}_{\text{baseline}}$	Training (s)	Test (s)
More data	87,650,897	1,696,802	92.13%	3,655.29	24.93
More features	10,000,001	11,799,345	75.14%	615.34	73.93

eling issues. It is interesting to see the performance gain by increasing the training size.

In Table 11, we compare the following two settings.

1. We consider a larger training set of 87M sold items. We use binary representation and normalize each instance to a unit vector.

2. We consider the 10M training set used in earlier experiments and apply bigram + unigram features.

For both settings, we train Crammer and Singer multi-class SVM. Table 11 indicates that the improvement by increasing training size is not that significant. Therefore, for product title classification, investigating various modeling issues may be a more effective way to improve the performance.

8.3 Implementation of One-versus-one Multi-class SVM

We discussed in Section 6 that for a k-class data, the one-versus-one method has a disadvantage of requiring $O(k^2n)$ space to store $k(k-1)/2$ model vectors, which is much larger than $O(kn)$ of the other two methods. For training 41M degree-2 polynomial features in Section 7, storing the model (assuming double precision) needs $41\text{M} \times 29 \times 28/2 \times 8 = 129\text{GB}$, which is beyond the capacity of our machine.

To reduce the memory consumption, we observe that many elements in model vectors are zeros and need not be stored. This situation occurs because some features of rare tokens are always zero in a pair of two classes. Actually, the model vectors for the one-versus-one method shown in Table 8 with poly2 features only have 350M non-zero elements, which can be stored in a sparse form. Then the memory for storing the model is reduced from 129G to less than 5G. This implementation makes the one-versus-one method possible for training large product title sets. Indeed, ours may be the largest one-versus-one SVM ever solved.

9 Conclusions and Future Issues

In this paper, we check the difference between product title classification and general text classification. It is observed that some conventional thoughts on text classification still apply to product title classification. For example,

data normalization is essential for faster SVM training. However, we find some interesting differences.

1. Stemming and stop-word removal should not be applied in product title classification. In contrast, for text classification, these procedures are sometimes beneficial.

2. Because product title lengths are similar, our analysis in Section 5 shows that if SVM is applied, data normalization does not downgrade the performance. Further, because words in a title are generally different, our derivation indicates that binary and TF-IDF representations perform similarly.

3. Because titles are often short, our analysis in Section 7 shows that bigram or degree-2 polynomial mappings are more effective than the case in general text classification.

In addition, we have a useful conclusion for general text classification. For optimization methods used in Section 6 to train one-versus-rest and Crammer and Singer SVM, if their numbers of iterations are similar, Crammer and Singer SVM is faster because of less frequent data accesses.

We believe these conclusions will be very useful for practitioners.

While we have conducted extensive experiments and analyses, this work is *far from* a complete study on product title classification. We list the following future issues.

1. Because titles are short and different sellers may submit similar items, a product title data set may contain duplicated instances. Then SVM problem (2) can be interpreted in a different way. If we identify unique instances $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\bar{l}}\}$, then the optimization problem can be written as

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{\bar{l}} C_i \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i), \tag{10}$$

   where $C_i = C \times (\#\text{occurrences of } \boldsymbol{x}_i)$. Therefore, for an instance appearing in the training set several times, C_i is larger and the classifier attempts to classify it correctly. It is interesting to check the performance without giving duplicated instances larger C_i values. That is, C_i in Eq. (10) is replaced by C. Evaluation is another issue because the test set may also contain duplicated instances.

   A related issue is data imbalance, although our data set is rather balanced (20 of 29 classes contain 1%–10% of training data). This issue is often application oriented, so suitable modeling techniques and evaluation criteria should be developed according to the business goal.

2. In our investigations in Sections 4–7, we sequentially apply settings concluded from earlier sections. For example, after Section 4, stemming is

not applied in all subsequent experiments. Therefore, we do not know, for example, if adding bigram features remedies the problem of stemming or not. Of course it is not possible to experiment with all settings. However, more future experiments help to make this study more complete.

3. Although we have considered only one large real-world set, we hope that our conclusions hold for other product title sets. We will experiment with other sets to confirm our findings. We mentioned in Section 6 that more investigations on the three multi-class strategies are needed. Obtaining results of other data sets can definitely help the analysis.

4. From Tables 8 and 11, on a single computer, our multi-class SVM code can train

   10M instances, 41M features in 25 minutes, and
   87M instances, 1.6M features in 60 minutes.

   This speed is quite remarkable. However, a direct use of LIBLINEAR fails on such large data because in most places it uses 32-bit integers to access data. We must modify some places to use 64-bit integers. How to extend LIBLINEAR or other text classification software to a truly 64-bit setting is a very important future direction.

Part of our experimental code, in particular, the extension of LIBLINEAR for one-versus-one multi-class SVM, will be publicly available.

References

[1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations*, vol. 11, 2009.

[2] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, 2009.

[3] S. T. Dumais and H. Chen, "Hierarchical classification of Web content," in *SIGIR*, 2000, pp. 256–263.

[4] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *ICML*, 1997.

[5] D. Shen, J. D. Ruvini, M. Somaiya, and N. Sundaresan, "Item categorization in the e-commerce domain," in *CIKM*, 2011, pp. 1921–1924.

[6] D. Zhang and W. S. Lee, "Question classification using support vector machines," in *SIGIR*, 2003, pp. 26–32.

[7] B. Qu, G. Cong, C. Li, A. Sun, and H. Chen, "An evaluation of classification models for question topic categorization," *Journal of the American Society for Information Science and Technology*, vol. 63, pp. 889–903, 2012.

[8] A. Khoo, Y. Marom, and D. Albrecht, "Experiments with sentence classification," in *Australasian Language Technology Workshop*, 2006.

[9] R. Bekkerman and M. Gavish, "High-precision phrase-based document classification on a modern scale," in *KDD*, 2011.

[10] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi, "Learning to classify short and sparse text & web with hidden topics from large-scale data collections," in *WWW*, 2008, pp. 91–100.

[11] M. Chen, X. Jin, and D. Shen, "Short text classification improved by learning multi-granularity topics," in *IJCAI*, 2011, pp. 1776–1781.

[12] B. Wolin, "Automatic classification in product catalogs," in *SIGIR*, 2002, pp. 351–352.

[13] E. Cortez, M. Rojas Herrera, A. S. da Silva, E. S. de Moura, and M. Neubert, "Lightweight methods for large-scale product categorization," *JASIST*, vol. 62, pp. 1839–1848, 2011.

[14] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22–31, 1968.

[15] M. Porter, "An algorithm for suffix stripping," in *Readings in information retrieval*, 1997, pp. 313–316.

[16] D. A. Hull, "Stemming algorithms – a case study for detailed evaluation," *JASIS*, vol. 47, pp. 70–84, 1996.

[17] E. Riloff, "Little words can make a big difference for text classification," in *SIGIR*, 1995, pp. 130–136.

[18] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–20, 1972.

[19] B. E. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT*, 1992.

[20] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *ECML*, 1998, pp. 137–142.

[21] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of IEEE*, 2012, to appear.

[22] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: a case study in handwriting digit recognition," in *ICPR*, 1994, pp. 77–87.

[23] D. D. Lewis, "An evaluation of phrasal and clustered representations on a text categorization task," in *SIGIR*, 1992.

[24] R. Bekkerman and J. Allan, "Using bigrams in text categorization," UMass Amherst, Tech. Rep. IR-408, 2004.

[25] D. Meretakis and B. Wüthrich, "Classification as mining and use of labeled itemsets," in *SIGMOD*, 1999.

[26] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *JMLR*, vol. 9, pp. 1871–1874, 2008.

[27] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[28] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," in *Neurocomputing: Algorithms, Architectures and Applications*, 1990.

[29] J. Weston and C. Watkins, "Multi-class support vector machines," in *ESANN*, 1999, pp. 219–224.

[30] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *JMLR*, vol. 2, pp. 265–292, 2001.

[31] Y. Lee, Y. Lin, and G. Wahba, "Multicategory support vector machines," *JASA*, vol. 99, pp. 67–81, 2004.

[32] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, "A sequential dual method for large scale multi-class linear SVMs," in *KDD*, 2008.

[33] M.-L. Antonie and O. R. Zaïane, "Text document categorization by term association," in *ICDM*, 2002, pp. 19–26.

[34] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear SVM," *JMLR*, vol. 11, pp. 1471–1490, 2010.

Appendix

The proof is straightforward by reformulating the optimization problem to have a new variable $\bar{\boldsymbol{w}} = \boldsymbol{w}/\Delta$.

$$\begin{aligned}
&\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l} \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0) \\
\equiv &\min_{\boldsymbol{w}} \Delta^2 \frac{1}{2}\|\frac{\boldsymbol{w}}{\Delta}\|^2 + C\sum_{i=1}^{l} \max(1 - y_i (\frac{\boldsymbol{w}}{\Delta})^T (\Delta \boldsymbol{x}_i), 0) \\
\equiv &\Delta^2 \left(\min_{\bar{\boldsymbol{w}}} \frac{1}{2}\|\bar{\boldsymbol{w}}\|^2 + \frac{C}{\Delta^2}\sum_{i=1}^{l} \max(1 - y_i \bar{\boldsymbol{w}}^T (\Delta \boldsymbol{x}_i), 0) \right).
\end{aligned}$$