A Self-Pruning Classification Model for News

Leonidas Akritidis, Athanasios Fevgas, Panayiotis Bozanis
Department of Electrical and Computer Engineering
University of Thessaly
Volos, Greece
Email: {leoakr,fevgas,pbozanis}@e-ce.uth.gr

Miltiadis Alamaniotis
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, USA
Email: Miltos.Alamaniotis@utsa.edu

***Abstract*—News aggregators are on-line services that collect articles from numerous reputable media and news providers and reorganize them in a convenient manner with the aim of assisting their users to access the information they seek. One of the most important tools offered by news aggregators is based on the classification of the articles into a fixed set of categories. In this article, we introduce a supervised classification method for news articles that analyzes their titles and constructs multiple types of tokens including single words and n-grams of variable sizes. In the sequel, it employs several statistics, such as frequencies and token-class correlations, to assign two importance scores to each token. These scores reflect the ambiguity of a token; namely, how significant it is for the classification of an article to a category. The tokens and their scores are stored in a support structure that is subsequently used to classify the unlabeled articles. In addition, we propose a dimensionality reduction approach that reduces the size of the model without significant degradation of its classification performance. The algorithm is experimentally evaluated by employing a popular dataset of news articles and is found to outperform standard classification methods.**

***Index Terms*—data mining, machine learning, supervised learning, classification, news**

I. Introduction

Nowadays, all reputable news providers maintain Web portals that are frequently updated with the latest news and events. The existence of these portals allowed a significant portion of the users to switch to on-line media to satisfy their informational needs. However, different reporters and journalists have different aspects on the same events, which, in turn, leads to the coverage of an event under multiple diverse points of view. This pluralism created the need for platforms that automatically collect news articles from different sources and group them by category and topic. These platforms are broadly known as news aggregation services, or *news aggregators*.

From the engineering perspective, news aggregators face multiple challenging problems, including the automatic and frequent collection of articles, the grouping of articles covering the same topic, their classification into a predefined set of categories, and so on. In this paper, we examine the last problem, namely, the classification of the news articles. Since many news sites exist and numerous articles get published at high rates, the issue of effective classification becomes crucial for assisting the users to locate the information they seek.

Classification is one of the most well-studied problems in the scientific field of supervised machine learning. The relevant literature contains a vast number of effective solutions, including statistical models (such as linear or logistic regression), data structures (such as decision trees and random forests), and neural networks. Another branch of the relevant research comprises works focusing on the problem of feature extraction; that is, the identification of the features that are the most important for classifying the data.

The specific problem of news articles classification has also attracted a considerable amount of relevant research. Some traditional classification methods – including Naive Bayes and Support Vector Machines (SVMs) – have been utilized in numerous prior works [1], [2]. Moreover, other tools developed deep learning solutions based on convolutional and recurrent neural networks [3], [4], [5]. Another family of methods employed the profiles of the users and classify the news articles according to their needs and interests [6], [7].

In this paper, we introduce a statistical method to confront this significant problem. In a spirit similar to some existing methods, our algorithm takes into consideration only the titles of the articles. More specifically, it sets a hyper-parameter N and computes the complete set of the $1, 2, \ldots, N$ -grams from the titles. These n-grams are then inserted in an assistant lexicon that also maintains some useful statistics, such as frequencies and lists of occurrences. Moreover, each entry is accompanied by an importance score that reflects its significance for the classification of an article. Next, n-grams and their statistics are used to create the training model.

Furthermore, notice that the computation and storage of numerous n-grams leads to a significant increase in the size of the aforementioned lexicon structure. Here we develop a strategy that prunes the records of this structure by examining their importance scores. In particular, it is assumed that a n-gram with a small importance score is also of small value; hence, it can be removed from the lexicon without a considerable loss in performance. As shown in the experiments, this approach reduces the dimensionality of the underlying feature space without suffering significant losses in the accuracy.

The rest of the paper is organized as follows: In Section II, we refer to the most important news classification methods, whereas in Section III, we describe the core elements of the proposed model, including the importance scores and the training and test phases. Finally, Sections IV and V contain the experimental valuation of our method and the conclusions of our work, respectively.

978-1-7281-4959-2/19/$31.00

II. Related Work

The increasing popularity of the on-line news portals has rendered the problem of automatic news articles classification particularly important. So, during the last years, numerous researchers proposed solutions with the aim of addressing it.

Some earlier works employed traditional supervised classification models such as naive Bayes, standard statistical methods, and maximum entropy [8], [1], [9]. Notice that these methods are closely related to specific languages, for example, Arabic and Indonesian. Moreover, in [10], a probabilistic analysis of the Rocchio relevance feedback algorithm is presented within a text categorization framework. In addition, the study of [2] employed Support Vector Machines (SVMs) to classify news into different groups in Twitter.

Due to its nature, the problem can also be tackled by utilizing text and short-text classification algorithms [11], [12]. The deep learning approaches now dominate this particular field of research, since their improved effectiveness attracted a large number of researchers. For instance, [3] modeled short texts by employing semantic clustering and neural networks. This algorithm initially employs a clustering algorithm that detects the meaningful multi-scale semantic units. These units are subsequently transfered to a convolutional layer, where a max-pooling operation is performed.

Moreover, [5] and [4] demonstrated the superiority of the character-level convolutional networks over some state-of-the-art models, such as word-based ConvNets, bag of words, and recurrent neural networks. Similarly, in [13] the authors showed the effectiveness of the simple convolutional neural networks with a little hyper-parameter tuning. This study also reported further gains in performance by employing task-specific vectors, with some additional and careful fine-tuning.

A significant portion of the relevant research studied the problem of news articles classification in a personalized manner. In these works, the primary objective is not to simply categorize the news articles into a predefined set of classes, but to provide a classification in combination with the profile of a user. The research of [6] uses a Bayesian classifier to select articles of interest to a specific user, according to his/her profile. Another method that classifies the news articles, by taking into account the emotions of the readers, is introduced in [7], whereas [14] introduces a personalized approach to the problem. More specifically, the authors propose the construction of search queries by using the user-define keywords. In the sequel, the classifier obtains both positive and negative training documents and employ SVMs to build the classification model.

The method of this article is based on the preliminary algorithm of [15] and its enhanced version of [16]. Both methods were proved effective in the classification of research papers and products, respectively. However, the former was not based on importance scores and lacked a built-in dimensionality reduction method. Compared to the second method, the news articles are different than products; hence the linguistic processing of their titles included additional steps such as stop-word removal and stemming (which also contributed to the reduction in the size of the model) and careful treatment of punctuation. Moreover, since this processing affects the length of a title, the importance scores here are also different.

III. News Articles Classification

This section describes in details the proposed method for news articles classification. The discussion begins with some necessary preliminary elements and proceeds with the presentation of the model, including its training and test phases.

A. Overview

Let us consider the set Y that contains the classes (or labels) where the articles will be classified into. Notice that Y can be either flat, or hierarchical; namely, it may include entirely unrelated classes, or categories that have parent-child relationships (i.e., categories with multiple subcategories).

In addition, we use $X = \{x_1, x_2, \ldots\}$ to denote the samples in the dataset, that is, the news articles. Each article is represented by its title τ, which includes a series of words $(w_1, w_2, \ldots, w_{l_x})$, where l_x denotes the title length (in words). Apart from its title, an article consists of its main body of text and some additional information, such as dates, reporter or journalist names, etc. However, this metadata is usually not consistent (i.e., it is not present in all articles), skewed, incomplete, erroneous, and in some cases, noisy. Therefore, in this article we do not take this metadata into consideration and we rely on the provided titles only.

In contrast, our method is based on the construction variable-sized n-grams. n-grams are sequences of n adjacent words within a string and they constitute a common technique for capturing the correlation between two or more successive words. They are computed by sliding a window of fixed length n over the examined string (in a left to right fashion).

In the following presentation, we use the term *tokens* to collectively refer to both n-grams and single words.

B. Classification Model

The approaches that build statistical models usually identify possible strong relationships between the words of a title and the provided categories. Examples of such methods include the well-known tf-idf scoring scheme, or other similar techniques.

Nevertheless, a single word may be simultaneously correlated with many classes (i.e., it may be ambiguous), or it may incorrectly indicate a strong correlation with a specific class. For example, the word *company* may be correlated with many categories, like *Economics*, *Politics*, or even *Society*. For this reason, single words are not adequate to identify the desired relationships and more complex representations are required.

To overcome this issue, apart from single words, we also utilize n-grams, which, in contrast to single title words, have a smaller probability of being ambiguous. In the aforementioned example, the 2-gram *company shares* clearly refers to *Economics*, whereas *company hires* is connected to *Society*. Intuitively, longer common n-grams are stronger indicators of the category that an article falls into.

The previous example demonstrated that the tokens of the title of an article may be highly related to one or more

categories on a different way. As stated earlier, a token t has its own *ambiguity* with respect to a class y. It follows that highly ambiguous tokens should be considered as less important during classification, compared to tokens having a strong correlation with a specific category. The importance of the tokens can be quantified by introducing an *importance score* I_t that is defined as follows:

$$I_t = l_t \log \frac{|Y|}{f_t} \quad (1)$$

where l_t is the length of t in words (recall that a token can be either a single word, or a n-gram). Moreover, f_t denotes the frequency of t, that is, the number of classes which are related to it. According to Eq. 1: i) a token that is correlated with many categories is considered more ambiguous and should be assigned a lower importance score, and ii) the importance of a token is proportional to its length.

Notice also that we do not simply construct fixed length n-grams, but tokens of variable sizes. Therefore, we preset a hyper-parameter N and we compute all the $1, 2, \ldots, N$ -grams. For instance, given that $N = 4$, the algorithm will create all the $1, 2, 3$, and 4 -grams. The extracted variable sized tokens are stored within a lexicon structure that, for each stored entry t, maintains:

- the aforementioned frequency f_t,
- its importance score I_t of Eq. 1, and
- a list Λ_t of $(y, f_{t,y})$ pairs, where y is a class, and $f_{t,y}$ is a counter which shows how many times t has been correlated with y.

In Subsection III-C we discuss the construction process of this data structure. In addition, in Subsection III-D we show how it is used to classify the articles in the dataset. Based on the concepts of [16], in Subsection III-E, we propose a pruning technique for reducing its dimensionality.

C. Training Phase

The training algorithm is outlined by the steps mentioned in Algorithm 1. The process begins with the initialization of an empty lexicon L that will be used to store the extracted tokens. Each article x in the training set is labeled by y and its title τ is tokenized. The extracted single words are then passed through a linguistics processor that performs a series of transformations, such as stop-words removal, stemming, and case folding. After this procedure, the n-grams are constructed by sliding a window of length n, over each title. Notice that in step 4 we construct variable sized n-grams by utilizing N such windows, with lengths ranging from 1 to N.

For each produced token, a look up in the lexicon L is executed (step 7). In case the look-up fails, an empty Λ_t list is initialized by inserting the pair $(y, f_{t,y})$ with $f_{t,y} = 1$ (steps 16–17). This pair reflects the relationship between t and y; it says us that t has been correlated with y one time. Then, t is inserted in L and its frequency f_t is set equal to 1 (steps 18–19). f_t reveals that t is correlated with one category.

On the other hand, if the look-up succeeds (t exists in L), then we immediately search for y within its Λ_t list to discover if t was previously associated with y. If this is the first co-occurrence of t and y (i.e., $y \notin \Lambda_t$), we increase f_t by 1 (step 11), and we insert the the pair $(y, f_{t,y})$ with $f_{t,y} = 1$ in Λ_t. In the opposite case, we only increase $f_{t,y}$ by 1 (step 13).

The algorithm ends by visiting each record t of the lexicon L. During each visit, we compute the importance score of each token by applying Eq. 1. We also sort the Λ_t lists of each token in decreasing $f_{t,y}$ order. This action will facilitate the application of our model pruning strategy, that is later described in Subsection III-E.

Algorithm 1: Training phase

```
1  initialize the token lexicon L;
2  for each article x with class y in the training set do
3      tokenize and process its title τ;
4      for each n ∈ [1, N] do
5          construct all tokens T_n of length n from τ;
6          for each token t ∈ T_n do
7              if t ∈ L then
8                  if y ∉ Λ_t then
9                      f_{t,y} ← 1;
10                     Λ_t.insert(y, f_{t,y});
11                     f_t ← f_t + 1;
12                 else
13                     f_{t,y} ← f_{t,y} + 1;
14                 end
15             else
16                 f_{t,y} ← 1;
17                 Λ_t.insert(y, f_{t,y});
18                 f_t ← 1;
19                 L.insert(t);
20             end
21         end
22     end
23 end
24 for each token t ∈ L do
25     I_t ← l_t log(|Y|/f_t);
26     sort Λ_t in decreasing f_{t,y} order;
27 end
```

D. Test Phase

This subsection describes the process of assigning labels to the unclassified articles by using the aforementioned model. The basic steps of this process are presented in Algorithm 2. The test phase begins by initializing an empty list of candidate classes Y' for each article x. Our goal is to populate this list with the most similar classes to x, and ultimately, to assign one of these classes to x. At next, the title τ of x is processed by applying the strategy we described in the training phase.

The tokens of τ (i.e. unigrams, bigrams, etc) are computed by employing the aforementioned sliding windows with lengths ranging from 1 to N. Each token $t \in \tau$ is subsequently searched within L; if it is found, its frequency f_t, the importance score I_t, and the entries of the Λ_t list are retrieved.

Algorithm 2: Test phase

```
for each unlabeled article x in the test set do
    initialize a list of candidates classes Y';
    tokenize and process its title τ;
    for each n ∈ [1, N] do
        construct all tokens T_n of length n from τ;
        for each token t ∈ T_n do
            if t ∈ L then
                for each pair (y, f_{t,y}) ∈ Λ_t do
                    if y ∉ Y' then
                        Y' ← Y' + y;
                        S_y ← f(I_{t,y}, Q_{t,y})
                    else
                        S_y ← S_y + I_{t,y}
                    end
                end
            end
        end
    end
    sort Y' in decreasing S_y order;
    y ← Y'[0];
end
```

The procedure continues by iterating through the entries of Λ_t (recall that Λ_t enlists $(y, f_{t,y})$ pairs) and for each $(y, f_{t,y})$ pair we look for y within the list of candidate classes Y'. In the case of an unsuccessful query, y is inserted in Y'. Moreover, a score $S_{t,y}$ for the candidate class y is calculated, with the respect to the importance score I_t of each involved token, and a quantity $Q_{t,y}$ which will be presented shortly. In the opposite case, the score S_y is updated. After the examination of all tokens of τ, the list of candidate classes Y' is sorted in decreasing score order. The article x is then assigned to the the category $Y'[0]$ that received the highest S_y score.

Regarding the computation of the $S_{t,y}$ scores, we apply a technique that is inspired from the ranking models of current Web search engines. More specifically, $S_{t,y}$ is determined by a linear combination of the 'query independent' importance scores I_t and a 'query dependent' term $Q_{t,y}$ according to the following equation:

$$S_{t,y} = k_1 I_t + k_2 Q_{t,y} \tag{2}$$

where k_1 and k_2 are two constants which that the constraint $k_1 + k_2 = 1$. Regarding the $Q_{t,y}$ factor, we employed a variant of the popular tf-idf scheme that has been proved quite effective in multiple machine learning and IR tasks. This variant leads to satisfactory classification performance and it is defined as follows:

$$Q_{t,y} = \log f_{t,y} \log \frac{|Y|}{f_t} \tag{3}$$

where $f_{t,y}$ is the aforementioned frequency value that stores the number of correlations of a token t with a category y.

E. Dimensionality Reduction & Model Pruning

The construction of multiple types of n-grams with variable lengths in L eventually leads to a significant increase in the total number of features used by the model. To improve the scalability of our solution, a dimensionality reduction technique is required. Here we present a method that achieves this requirement, whereas it simultaneously has a small impact on the classification accuracy of the algorithm.

The proposed reduction method exploits the importance scores $I_{t,y}$ of the tokens. In particular, it establishes a predefined *cut-off threshold* C and excludes all the tokens whose importance score is lower than C. These tokens are assumed to be of small, or no informational value and are effectively pruned from the lexicon L. C is defined as a percentile T of the highest importance score in the lexicon as indicated by the the following formula:

$$C = T \max_{t,y} I_{t,y} \tag{4}$$

In our experiments we employed various values of T; our evaluation illustrates how T reduces the model size in combination with the classification performance.

IV. Experiments

In this section, we experimentally evaluate the proposed method in terms of classification accuracy and size of the trained model. For the purpose of our experiments, we used the UCI News Aggregator dataset that consists of about 400,000 news stories and is publicly available from Kaggle[1]. The articles are accompanied by their title and a class label that falls into one of the following four categories: *business*, *science and technology*, *entertainment*, and *health*. In addition, we used the 70% of the articles to construct our training set, whereas the rest 30% were left for testing purposes.

In the following analysis, we refer to the proposed algorithm by using the abbreviation *SNC (Supervised News Classifier)*. SNC is compared against the well-known Logistic Regression algorithm (abbreviated as LogReg).

A. Classification Performance

Figure 1 depicts the accuracy of SNC in comparison with Logistic Regression for two feature spaces: i) $N = 1$, where the space includes only the single words of the titles (i.e., no n-grams were created), and ii) $N = 3$, that contains all the unigrams, bigrams and trigrams. For these two feature spaces, the classification accuracy of all methods was measured for various levels of model pruning. More specifically, we altered the value of the cut-off threshold T of Eq. 4 in the range [0.0,1.0], by taking steps of 0.1 each time. To perform a fair comparison, LogReg was fed with the same set of features (i.e., n-grams) and with the same importance scores as weights.

For the first feature space ($N = 1$, left diagram of Figure 1) and with all unigrams present (i.e., $T = 0$), SNC had a

[1]https://www.kaggle.com/uciml/news-aggregator-dataset/downloads/news-aggregator-dataset.zip/1

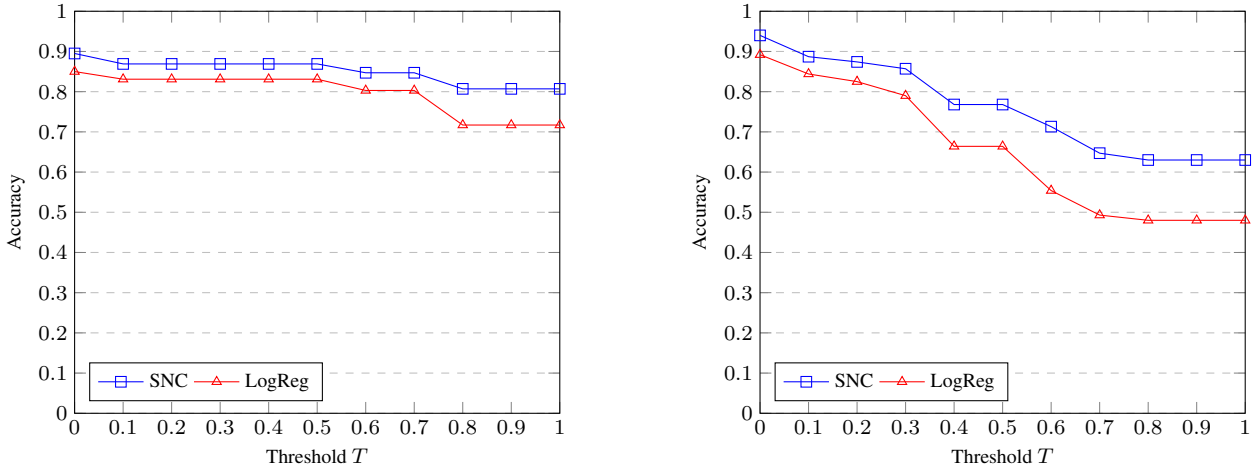

Fig. 1. Comparison of the classification performance of SNC against Logistic Regression (LogReg) for i) $N = 1$ (left), and ii) $N = 3$ (right).

satisfactory classification accuracy of 89.5%. For values of T ranging from 0.1 to 0.5, the accuracy dropped slightly to 0.869, that is, by a margin of 3%. After these values, the accuracy falls at higher rates. Regarding the adversary algorithm, the accuracy of LogReg started from 85% (for $T = 0$) and it degraded in a fashion similar to SNC as T increased.

The highest performance for SNC was achieved for $N = 3$ and $T = 0$; its accuracy in this scenario reached 94%. The accuracy falls slowly for threshold values ranging from 0.1 to 0.3; for $T = 0.3$ it is measured at 86%, 8% lower than the maximum. In the next subsection, we show that for $T = 0.3$, the model includes 16 times fewer tokens and 7.5% fewer Λ_t entries compared to the initial model size (i.e., for $T = 0$). Consequently, by pruning a huge amount of the model data we suffer only a small loss of about 8% of its classification accuracy. The accuracy for LogReg in this case was measured at 89.2%, significantly lower than the one of SNC.

Moreover, both algorithms perform better in the large feature space ($N = 3$), but only for small values of the cut-off threshold, that is $T \leq 0.3$. Apparently, above this value of T, numerous important bigrams and trigrams are removed from the model and this loss leads to a degraded performance.

Notice that these results are considerably different than the ones reported in [16]. In that work, which examined the problem of products classification, it was found that by pruning almost half of the model's data, the decrease in performance was only 2.5%. In this problem, the loss in accuracy is slightly greater, however, the volume of the pruned data is much larger.

B. Model Size

In Table I reports indicative model sizes with respect to the value of n. For the single-word dimensional space (i.e. $n = 1$), the lexicon L enlists 46273 words. The number of correlations of these words with the 4 classes (that is, the number of $(y, f_{t,y})$ pairs within all Λ_t lists) was 75370. This is translated to an average correlation of a single word with 1.63 categories. For $n = 2$, 739 thousands of bigrams were extracted from the titles of the articles, whereas their Λ_t lists stored approximately 863 thousands items; hence, each 2-gram was associated on average with only 1.16 categories. Similarly, the 3-grams are correlated on average with 1.09 classes. These values confirm of our main assumption where we predicted that the length of a token is tightly connected to its ambiguity. In other words, greater importance scores should be assigned to longer tokens.

At next, we evaluate our proposed dimensionality reduction method. Similarly to the previous experiment, we examine two feature spaces (for $n = 1$ and $n = 3$), and we modify the cut-off threshold T in the range $[0.0, 1.0]$. Figure 2 illustrates the decrease in the number of tokens and the number of Λ_t entries stored within L, as T increases. Notice that, for $T = 0$, the sizes are identical to the ones that we report in Table I.

For $n = 1$, the left diagram of Figure 1 shows that at $T = 0.6$ the accuracy starts to decrease considerably (by a margin of 6%). At this point, Figure 2 tells us that the model includes 8792 tokens and 31470 Λ_t entries. Compared to the initial model size (46273 tokens and 75370 Λ_t entries), we achieve a reduction of about 5.26 times in the number of tokens and 2.4 times in the number of Λ_t entries. Hence, large savings lead to only a small loss in performance. Similar savings are also achieved for $N = 3$.

TABLE I
INITIAL SIZES OF L WITH RESPECT TO THE VALUE OF n.

n	Tokens	Λ_t pairs
1 (words)	46273	75370
2 (bigrams)	739346	863431
3 (trigrams)	1428174	1554562
1, 2	785619	938801
1, 2, 3	2213793	2417993

V. CONCLUSIONS

In this paper, we examined the problem of news articles classification that is considered crucial for news aggregators and portals. In general, the effective classification can enhance the quality of the provided information and assists users to

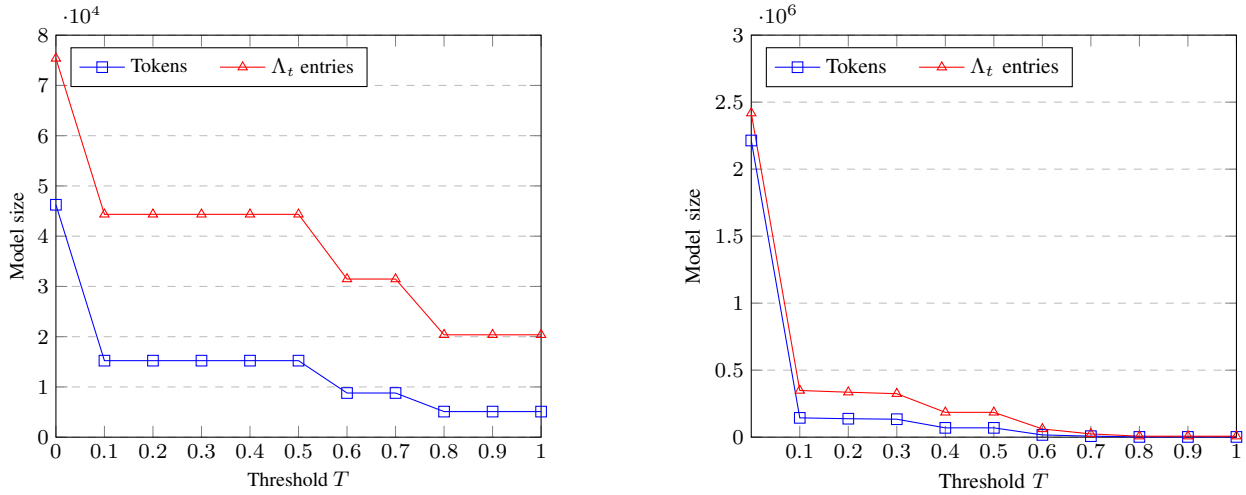

Fig. 2. Model size (overall number of tokens and Λ_t entries) reduction with respect to the cut-off threshold T for $N = 1$ (left), and $N = 3$ (right).

easily locate their desired content. The proposed method is based on several key observations including:

- The words of a title are usually correlated with multiple categories, consequently, they present a level of ambiguity. This ambiguity represents how significant a correlation between a token and a class is.
- Sequences of adjacent words (i.e. n-grams) have a smaller probability of being ambiguous; the longer the sequence is, the smaller ambiguity it presents.
- The ideal length of n-grams cannot be easily determined and, to maximize the performance, multiple types of n-grams of variable sizes must be constructed.

These observations provided the theoretical basis of the proposed model. More specifically, after the tokenization and the morphological analysis of the titles of the articles, the generated tokens (single words and n-grams) are stored in an assistant lexicon structure L. Each token $t \in L$ is assigned an importance score that reflects its global importance. Furthermore, for each class y, we also define another score that reveals how strong the correlation between t and y is.

The construction of variable length tokens leads to increased lexicon sizes, which in turn affects scalability. In this work, we also introduced a dimensionality reduction method based on the importance scores of the tokens. This method identifies the unimportant tokens and their occurrences, and it effectively excludes them from the model.

The proposed algorithm was experimentally evaluated by using a real-world news dataset. The results revealed that by extracting $1, 2$ and 3-grams from the titles, the classification accuracy reached a remarkable 95% and outperformed Logistic Regression by a significant margin. Regarding the built-in dimensionality reduction method, the experiments have shown that by pruning almost half of the data stored in the model, accuracy was dropped by an infinitesimal margin of only 2%.

References

[1] A. D. Asy'arie and A. W. Pribadi, "Automatic news articles classification in indonesian language by using naive bayes classifier method," in *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, 2009, pp. 658–662.

[2] I. Dilrukshi, K. De Zoysa, and A. Caldera, "Twitter news classification using svm," in *Proceedings of the 8th International Conference on Computer Science & Education*, 2013, pp. 287–291.

[3] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic Clustering and Convolutional Neural Network for Short Text Categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2015, pp. 352–357.

[4] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, vol. 333, 2015, pp. 2267–2273.

[5] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[6] R. Carreira, J. M. Crato, D. Gonçalves, and J. A. Jorge, "Evaluating adaptive user profiles for news classification," in *Proceedings of the 9th International Conference on Intelligent User Interfaces*, 2004, pp. 206–212.

[7] K. H.-Y. Lin, C. Yang, and H.-H. Chen, "Emotion classification of online news articles from the reader's perspective," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008, pp. 220–226.

[8] S.-B. Kim, H.-C. Seo, and H.-C. Rim, "Poisson naive bayes for text classification with feature weighting," in *Proceedings of the 6th International Workshop on Information Retrieval with Asian languages-Volume 11*, 2003, pp. 33–40.

[9] H. Sawaf, J. Zaplo, and H. Ney, "Statistical classification methods for arabic news articles," *Proceedings of Natural Language Processing*, 2001.

[10] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," Carnegie-Mellon University, Tech. Rep., 1996.

[11] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*, 2012, pp. 163–222.

[12] G. Song, Y. Ye, X. Du, X. Huang, and S. Bie, "Short text classification: A survey," *Journal of multimedia*, vol. 9, no. 5, p. 635, 2014.

[13] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1746–1751.

[14] C.-H. C. A. S. Ee and P. Lim, "Automated online news classification with personalization," in *4th international conference on asian digital libraries*, 2001.

[15] L. Akritidis and P. Bozanis, "A supervised machine learning classification algorithm for research articles," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 115–120.

[16] L. Akritidis, A. Fevgas, and P. Bozanis, "Effective product categorization with importance scores and morphological analysis of the titles," in *Proceedings of the 30th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 213–220.