World Wide Web (2016) 19:887–920
DOI 10.1007/s11280-015-0365-x

Graph vs. bag representation models for the topic classification of web documents

George Papadakis[1] · George Giannakopoulos[2] · Georgios Paliouras[2]

Received: 23 December 2014 / Revised: 19 May 2015 /
Accepted: 20 July 2015 / Published online: 12 August 2015

Abstract Text classification constitutes a popular task in Web research with various applications that range from spam filtering to sentiment analysis. In this paper, we argue that its performance depends on the quality of Web documents, which varies significantly. For example, the curated content of news articles involves different challenges than the user-generated content of blog posts and Social Media messages. We experimentally verify our claim, quantifying the main factors that affect the performance of text classification. We also argue that the established bag-of-words representation models are inadequate for handling all document types, as they merely extract frequent, yet distinguishing terms from the textual content of the training set. Thus, they suffer from low robustness in the context of noisy or unseen content, unless they are enriched with contextual, application-specific information. In their place, we propose the use of *n-gram graphs*, a model that goes beyond the bag-of-words representation, transforming every document into a graph: its nodes correspond to character or word n-grams and the co-occurring ones are connected by weighted edges. Individual document graphs can be combined into class graphs and graph similarities are employed to position and classify documents into the vector space. This approach offers two advantages with respect to bag models: first, classification accuracy increases due to the contextual information that is encapsulated in the edges of the n-gram graphs. Second,

✉ George Papadakis
gpapadis@di.uoa.gr

George Giannakopoulos
ggianna@iit.demokritos.gr

Georgios Paliouras
paliourg@iit.demokritos.gr

[1] Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilissia, 15784 Athens, Greece

[2] National Center for Scientific Research “Demokritos”, Patriarchou Grigoriou 27, Agia Paraskevi, 15310 Attica, Greece

it reduces the search space to a limited set of robust, endogenous features that depend on the number of classes, rather than the size of the vocabulary. Our thorough experimental study over three large, real-world corpora confirms the superior performance of n-gram graphs across the main types of Web documents.

Keywords Text classification · N-gram graphs · Web document types

1 Introduction

Text classification, also known as *text categorization*, is the task of automatically detecting one or more predefined categories that are relevant to a specific document [36, 37]. This process is typically carried out with the help of supervised machine learning techniques: a classification algorithm is trained over a corpus of labelled documents and captures the most distinguishing category patterns that can be used to classify the new, unlabelled instances. Text classification constitutes a popular research topic, due to its applicability to most kinds of Web documents, such as filtering spam out of e-mails [24], categorizing Web pages hierarchically [10] and analysing the sentiment of Social Media content [31]. As a result, its performance is critical for a wide range of tasks on the Web.

The most critical component of text classification is the *representation model* that converts every document into features fed to classification algorithms [13, 15, 26]. The goal of this paper is to examine the extent to which its functionality is affected by the type of document, given that not all Web documents are of the same quality. Unlike the clean, curated content of news articles or scientific publications, the user-generated content (UGC) that is posted on-line through Web 2.0 tools is excessively noisy. It actually poses the following serious challenges to the functionality of representation models:

(Ch1) *Multilinguality.* Representation models usually need to be fine-tuned to the language at hand in order to ensure high performance. This is typically done through *language-specific* pre-processing techniques, such as lemmatization, stemming and word-sense disambiguation with the help of dictionaries (e.g., WordNet[1]) [29]. UGC, though, can be written in any language – even in multiple languages. Typically, it lacks any information about the underlying language(s), unless a specialized technique for automatic language identification is used. To avoid the complexity and the overhead of these methods, text classification over UGC requires *language-neutral* representation models.

(Ch2) *Syntactical and grammatical errors.* UGC is particularly noisy, due to its casual, spontaneous writing and its minimal curation. Web users frequently participate in chats, posting their messages as fast as possible, without verifying their grammatical or spelling correctness. The resulting noisy content degrades the performance of representation models, hampering the identification of patterns in the form of repeated occurrences of the same characters or *tokens* (i.e., strings of characters that pertain to a specific term). Hence, UGC calls for representation models that are *robust to noise*.

(Ch3) *Sparseness.* A large part of UGC comprises free-form text that is rather short in length, due to size limitations. All messages in Twitter,[2] for instance, are restricted

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

[2] https://twitter.com

to 140 characters. These messages merely consist of a handful of words, thus lacking distinguishing information that can be used by representation models as evidence for identifying their topic. Their sparseness calls for *contextual* models that are able to extract more endogenous information under these settings.

(Ch4) *Evolving, non-standard vocabulary.* A large part of UGC pertains to informal communication between friends, who typically employ a casual "communication protocol" (e.g., slang words and dialects) [11]. The limited size of their messages also urges them to shorten words into neologisms that bear little similarity to the original ones (e.g., "gr8" instead of "great"). These phenomena restrict the applicability and degrade the performance of language-specific, dictionary-based pre-processing techniques. Instead, they call for *language-neutral*, *dictionary-free* representation models.

In this paper, we argue that the quality of Web documents affects the performance of text classification to a considerable extent. To assess this effect, we analyse the endogenous characteristics of Web documents, identifying four decisive criteria: (i) their length in characters or tokens, (ii) the diversity of their vocabulary, (iii) the noise they contain, and (iv) the frequency of special notation, such as URLs and hashtags. Based on these characteristics, we distinguish Web documents into three types – *curated*, *semi-curated* and *raw* ones – and argue that they induce significant variation in the effectiveness and the time efficiency of the representation models that typically lie at the core of text classification systems.

In more detail, the established approaches rely on *character* or *token n-grams* and are collectively called *bag models* [22, 36]. Essentially, they associate individual documents and topics with the frequent and discriminative characters or words that appear in them. However, they are inadequate for dealing with the above four challenges. Due to challenges Ch2 and Ch4, they yield numerous features that correspond to a huge search space, a phenomenon called *"curse of dimensionality"*. The resulting feature vectors are hard to fine-tune and pose serious limitations to the robustness of bag models, as they exclusively consider those features that have been extracted from the training set. This shortcoming is typically resolved through the incorporation of contextual, exogenous information, an approach that is application-specific [36]. Finally, the accuracy of bag models is restricted, because they disregard the valuable information that is contained in the sequence of n-grams in text. In the multilingual settings of UGC, this information is valuable in dealing with sparseness (challenges Ch1 and Ch3).

To overcome these drawbacks, we propose the use of a language-neutral representation model that is inherently robust to noise and takes contextual information into account: the *n-gram graphs*. It goes beyond the bag models by representing individual documents and entire topics as graphs. Their nodes correspond to character or token n-grams, while their weighted edges denote the frequency of co-occurrence of the adjacent nodes. In this way, the graph models enhance the traditional bag ones with contextual information that addresses sparseness, thus achieving higher effectiveness (i.e., classification accuracy).

Additionally, we explain how n-gram graphs can be used for text classification in a way that successfully addresses the curse of dimensionality, limiting the search space in order to ensure high efficiency: each labelled document of the training set is transformed into a graph and the graphs belonging to the same category are merged into a class graph; to classify an unlabelled document, we compare its n-gram graph with every class graph and estimate the numeric value of three graph similarity metrics. As a result, the dimensionality of the feature vector representing every document depends on the number of topics, rather than the vocabulary size. We also introduce a discretization approach for converting the originally numeric features into nominal ones. In combination, these two types of feature form hybrid approaches of high robustness at the cost of a slightly larger feature space.

The high performance of *n*-gram graphs is verified through a thorough experimental study on three large-scale, real-world corpora – one for each of the main document types. Its goal is to highlight the different challenges involved in each document type and how they affect the performance of representation models, especially that of *n*-gram graphs. We minimize language-dependency, assuming a *multi-lingual setting* that is important for many contemporary applications. We do not use any external knowledge and show that, under these conditions, the graph representation models provide better results than the traditional bag models in a variety of settings.

In summary, the main contributions of this paper are the following:

1. We introduce four content-based criteria for assessing the quality of Web documents. Based on them, we categorize Web documents into three main types, explaining their differences and how they affect the performance of text classification.
2. We explain how the *n*-gram graphs can be employed as a representation model for text classification and elaborate on its advantages over the bag models with respect to effectiveness and efficiency. We also introduce a discretization process for converting the numeric features of *n*-gram graphs into nominal ones and demonstrate that the combination of both feature types leads to a more robust performance.
3. We conduct a detailed experimental study over three large-scale, real-world datasets – one for each type of Web document. Based on the outcomes of our study, we fine-tune the internal parameters of the graph models and verify that they significantly outperform the bag ones with respect to classification accuracy.

The rest of the paper is structured as follows: in Section 2, we formally define the problem that we are studying and in Section 3, we analyse the *n*-gram graph model, comparing it with the traditional bag models. Section 4 presents our typology of Web documents, while Section 5 presents and discusses our experimental results. We present the most important related works in Section 6 and we conclude the paper in Section 7 along with directions for future work.

2 Problem definition

Related literature distinguishes text classification into several sub-problems, with *single-label text classification* being one of the most general and popular ones. It involves disjoint categories (i.e., each document is assigned to a single class) and lies at the core of many text classification applications. Prominent examples are text filtering, which distinguishes documents into relevant and irrelevant ones, and its most popular instantiation, namely spam detection [36]. For this reason, we exclusively consider single-label text classification in the following.

Among its applications, we focus on *topic classification* (TC) and employ it as a use case for illustrating the qualitative and quantitative differences between the various document representation models. Topic classification is the task of categorizing a given set of documents into thematic categories and constitutes a crucial process for many Web applications, ranging from news services to blogs and Social Media [26]. More formally, the task that we consider in this work can be defined as follows [36]:

Problem 1 (Single-label Topic Classification) Given a corpus of documents $\mathcal{D}$, a set of topics $\mathcal{T}$, and a set of training pairs $D_{tr} = \{< d_i, t_i >: d_i \in \mathcal{D} \wedge t_i \in \mathcal{T}\} \subset \mathcal{D} \times \mathcal{T}$, we seek a function $f : \mathcal{D} \rightarrow \mathcal{T}$, that minimizes the size $|E|$ of the set of false pairs (i.e., errors): $E = \{< d_i, t_i >: d_i \in \mathcal{D} \wedge f(d_i) \neq t_i\}$.

Given that we use a subset of the full corpus as a training set, we may fail to find the optimal function f and, thus, we are rather looking for the best approximation to it. In this effort, we exclusively consider *content-based representation models*, i.e., models that rely on the endogenous textual information of the given corpus. This means that we completely disregard *context-aware representation models*, which enhance TC through the incorporation of contextual information, such as related Web resources and special-purpose metadata (e.g., publication date). There are several reasons for this decision; contextual information may involve high extraction cost and usually comes in the form of text, thus calling for content-based representation models, as well. Most importantly, though, it constitutes an application-dependent parameter [36]. Given that we do not aim at optimizing the performance of a specific application, there is no need to consider such information in our analysis.

Instead, our goal is to examine the effect of challenges Ch1 to Ch4 on the performance of document representation models and to identify the model that adds more value to the basic textual patterns formed in the documents. In our analysis, we consider two types of content-based model: the bag and the graph ones. We place more emphasis on the latter category, examining the models that it generates with respect to the following aspects of TC: *How can we use the graph representation models in conjunction with existing machine learning techniques? Which configuration offers the best trade-off between the efficiency and the effectiveness of TC over Web documents? Is the accuracy and the speed of classification the same across different types of document (e.g., the curated content of news articles and the noisy content of Social Media)?* The conclusions of our study are expected to be directly applicable to other applications of text classification, as well.

3 Document representation models

In this section, we delve into the main content-based representation models, placing particular emphasis on their parameters. We begin with the presentation of the simple bag models in Section 3.1 and continue with the introduction of the more complex graph models in Section 3.2. We conclude their description in Section 3.3 with an analysis of their advantages and disadvantages when applied to Web documents.

3.1 Bag models

These models represent each document as a collection of features that correspond to the dimensions of a vector space. These features typically come in two forms:

- *character n-grams*, which are sequences of letters of length n, and
- *token n-grams*, which are sequences of words (i.e., tokens) of length n.

For both types of feature, we call n the *core size* of the corresponding bag model. For character-based bag models, n is usually set to 2, 3 or 4, and the resulting models are called *character bigrams* (C2G), *character trigrams* (C3G) and *character four-grams* (C4G), respectively. For token-based bag models, n is usually set equal to 1, 2 or 3, and the resulting models are called *token unigrams* (T1G), *token bigrams* (T2G) and *token trigrams* (T3G), respectively. To illustrate the functionality of bag models, consider the phrase "home_phone"; T1G represents it as $\{home, phone\}$ and C4G as $\{home, ome\_, me\_p, \_pho, phon, hone\}$.

The values of the features can be calculated in several different ways. The most common are [29]:

– a *binary* value that indicates the existence or absence of the corresponding n-gram in the text,
– a *numeric* value that corresponds to the (normalized) *Term Frequency* (TF) of the n-gram (i.e., how many times it appears in a given document), and
– a *numeric* value that corresponds to the *Term Frequency-Inverse Document Frequency* (TF-IDF) of the n-gram. This weighting scheme improves on the previous one by incorporating the reciprocal frequency of a word over the entire corpus (IDF) in order to reduce the impact of particularly common features (e.g., stop words).

To transform a document collection $\mathcal{D}$ into a bag model, we first need to identify the set of features $\mathcal{F}$ that define the vector space, i.e., the distinct character or token n-grams that appear in the documents of $\mathcal{D}$. Each document $d_i \in \mathcal{D}$ is then represented as a **document vector** $v_{d_i} = (v_1, v_2, \ldots, v_{|\mathcal{F}|})$, whose j^{th} dimension v_j corresponds to the value of the j^{th} feature for d_i [35]. Similarly, a topic $\mathcal{T}_j$ is modelled as a **class vector** ($v_{\mathcal{T}_j}$), whose dimensions comprise the aggregate values of the individual document vectors [29].[3]

3.2 Graph models

The *n-gram graph model* was introduced in [17] as a summary evaluation method. The motivation for it is to take into account the order of appearance of token and character n-grams in the original text. For example,[4] the phrases *China sues France* and *France sues China* have identical T1G representations, despite their inverse meaning; inevitably, both documents would be considered relevant to the class c = *legal actions brought by France*, yielding a false positive for the first phrase. To overcome this problem, the n-gram graph models extend the bag ones by associating neighbouring pairs of n-grams with edges that denote their frequency of co-occurrence.

The structure of n-gram graphs is formally defined as follows [17]:

Definition 1 An ***n*-gram graph** is an undirected graph $G = \{V, E, W\}$, where V is the set of its vertices, with each vertex corresponding to a distinct n-gram, E is the set of edges between co-occurring n-grams, and W is a function that determines the weight of each edge according to the co-occurrence frequency of its adjacent vertices. Each vertex is labelled by the corresponding n-gram and each edge by the labels of its adjacent vertices – concatenated in alphabetic order.

Similar to the bag models, the graph ones consider either n consecutive letters (i.e., *character n-gram graphs*) or n consecutive words (i.e., *token n-gram graphs*). To illustrate their functionality, consider the character trigram graph (i.e., $n = 3$)

[3] An alternative approach to forming a class vector is to extract the centroid from the vectors of the individual documents it comprises [29].

[4] Example borrowed from [29].

of Figure 1, which models the phrase "home_phone". Due to the association of n-grams, it captures more information than the corresponding bag representation of C3G: $\{hom, ome, me\_, e\_p, \_ph, pho, hon, one\}$.

To represent a document d_i, we create a **document graph** G_{d_i} by running a window of n characters or tokens over its textual content, breaking it into overlapping n-grams. Any two n-grams that are found within a certain distance are connected with an edge $e \in E_{d_i}$, whose weight denotes their frequency of co-occurrence in the entire document (two n-grams are considered neighbouring if their distance is lower than n steps [17]). In this way, each document is transformed into a graph that captures the contextual information of its co-occurring n-grams.

The same representation applies to a collection of documents that belong to the same topic. In this case, however, the graph is formed by merging the individual document graphs into a single **class graph** through the *update operator* [18]. This procedure operates as follows: given a collection of documents belonging to topic $\mathcal{T}_j$, an initially empty graph $G_{\mathcal{T}_j}$ is built; the $i\text{-}th$ document $d_i \in \mathcal{T}_j$ is then transformed into the document graph G_{d_i} that is merged with $G_{\mathcal{T}_j}$ to form a new graph $G_u = (V_u, E_u, W_u)$, where $V_u = V_{\mathcal{T}_j} \cup V_{d_i}$, $E_u = E_{\mathcal{T}_j} \cup E_{d_i}$ and $W_u(e) = W_{\mathcal{T}_j}(e) + (W_{d_i}(e) - W_{\mathcal{T}_j}(e)) \times 1/i$. Hence, the nodes and the edges of a class graph comprise the union of the nodes and edges of the individual document graph, while the weights of its edges incrementally converge to their overall average value, due to the division with i in the formula of $W_u(e)$. In this way, class graphs capture patterns common in the content of the entire topic.

3.2.1 Graph similarity metrics

To estimate the similarity between a document graph G_{d_i} and a class graph $G_{\mathcal{T}_j}$, we employ the following graph similarity metrics [17]:

(i) The *Containment Similarity* ($CSim$) expresses the proportion of edges shared among G_{d_i} and $G_{\mathcal{T}_j}$ and is formally defined as follows:

$$CSim(G_{d_i}, G_{\mathcal{T}_j}) = \frac{|E_{d_i} \cap E_{\mathcal{T}_j}|}{\min(|E_{d_i}|, |E_{\mathcal{T}_j}|)},$$

where $|E_x|$ denotes the number of edges in E_x (i.e., the *size* of the corresponding n-gram graph, G_x).

(ii) The *Value Similarity* ($VSim$) extends $CSim$ to take edge weights into account, as well. In this measure, each matching edge e with a weight $W(e, G_{d_i})$ in graph G_{d_i}

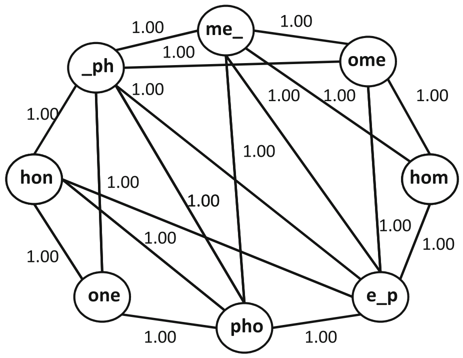

Figure 1 Character trigram graph (C3GG) for "home_phone"

and a weight $W(e, G_{\mathcal{T}_j})$ in graph $G_{\mathcal{T}_j}$ contributes $\mathrm{VR}(e)/\max(|G_{d_i}|, |G_{\mathcal{T}_j}|)$, where VR(e) stands for the *value ratio*, i.e., a symmetric, scaling factor that takes values in the interval [0, 1] and is defined as $\mathrm{VR}(e) = \frac{\min(W(e,G_{d_i}),W(e,G_{\mathcal{T}_j}))}{\max(W(e,G_{d_i}),W(e,G_{\mathcal{T}_j}))}$. Note that $\mathrm{VR}(e)$ amounts to 0 for the non-matching edges, i.e., for $e \notin (E_{d_i} \cap E_{\mathcal{T}_j})$. Putting all these together, we have:

$$VSim(G_{d_i}, G_{\mathcal{T}_j}) = \frac{\sum_{e\in E_{d_i}} \frac{\min(W(e,G_{d_i}),W(e,G_{\mathcal{T}_j}))}{\max(W(e,G_{d_i}),W(e,G_{\mathcal{T}_j}))}}{\max(|E_{d_i}|, |E_{\mathcal{T}_j}|)}.$$

$VSim$ converges to its maximum value $VSim_{max} = 1$ for graphs that share both the edges and the corresponding weights (i.e., $VSim_{max}$ indicates a perfect match between the compared graphs).

(iii) The *Normalized Value Similarity* ($NVSim$) is a variant of $VSim$ that factors out the relative size of the two graphs. Formally, it is defined as follows:

$$NVSim(G_{d_i}, G_{\mathcal{T}_j}) = \frac{\sum_{e\in E_{d_i}} \frac{\min(W(e,G_{d_i}),W(e,G_{\mathcal{T}_j}))}{\max(W(e,G_{d_i}),W(e,G_{\mathcal{T}_j}))}}{\min(|E_{d_i}|, |E_{\mathcal{T}_j}|)}.$$

This normalization is particularly important, since $VSim$ takes values very close to 0, when the class graph is much larger than the document graph.

From a different viewpoint, $CSim$ quantifies the co-occurrence of identical substrings in the compared documents. In this respect, it is related to the cosine similarity between bag n-gram models with binary weights. The main difference, though, is that $CSim$ considers the co-occurrence of pairs of n-grams (i.e., edges) instead of the occurrence of individual n-grams. Similarly, $VSim$ and $NVSim$ take into account the frequency of co-occurring n-grams and, thus, are analogous to the cosine similarity between frequency-based vectors of the bag n-gram models. Again, $VSim$ and $NVSim$ operate on pairs of n-grams, instead of considering the occurrence of individual n-grams.

3.2.2 *Classification with n-gram graphs*

The following procedure is used to train a classification algorithm on a collection of *labelled* documents $\mathcal{D}_l$ using the n-gram graphs model.[5] First, we build the class graphs $G_{\mathcal{T}_1}$, $G_{\mathcal{T}_2}$,...,$G_{\mathcal{T}_N}$, where N is the number of distinct topics contained in $\mathcal{D}_l$. Note that only a part of documents, specified by the "merge portion" parameter (see below, Section 3.2.3), participates in the creation of each class graph. Second, we transform each labelled document $d_i \in \mathcal{D}_l$ into a feature vector as follows:

1. We build the corresponding document graph G_{d_i}.
2. We compare G_{d_i} with the N class graphs, extracting the values of the aforementioned graph similarity metrics (i.e., $CSim$, $NVSim$ and $VSim$).
3. We put together all the derived similarities into a feature vector with $3 \times N$ dimensions that can be used as input to any classification algorithm.

[5]The implementation of this procedure in Java is provided publicly through the "Text Representation Models" project of Sourceforge.net at: http://sourceforge.net/projects/textmodels.

These three steps, which are illustrated in Figure 2, apply to *all* documents in $\mathcal{D}_l$, even those that have been included in the class graphs. The resulting feature vectors along with the ground-truth are used to train the algorithm.

To classify a collection of *unlabelled* documents $\mathcal{D}_u$ with the learned model, we simply apply steps 1 to 3 to all documents $d_i \in \mathcal{D}_u$ using the same class graphs ($G_{\mathcal{T}_1}$, $G_{\mathcal{T}_2}$,…,$G_{\mathcal{T}_N}$). The resulting feature vectors are then fed to the trained algorithm to drive its decision.

3.2.3 Configuration parameters

The n-gram graph model is quite versatile and adaptable to a variety of classification settings. This is ensured through four internal parameters that configure its performance:

1. The *feature type* specifies whether the classification features take numeric or nominal values or both of them.
2. The *model granularity* specifies whether the n-grams represent consecutive characters or tokens.
3. The *core size* determines the length of n-grams.
4. The *merge portion* determines the part of labelled documents that are involved in the creation of each class graph.

We elaborate on these parameters in the following and we experimentally investigate their effect on classification accuracy and time in Section 5. Note that the parameters "feature type" and "merge portion" are intrinsic to the graph models, while the parameters "model granularity" and "core size" apply to bag models, as well.

(i) *Feature type.* In the above section, we explained how we can extract feature vectors from the graph models. Their dimensions are exclusively *numeric*, as they correspond to graph similarity values. This approach ensures high time efficiency, due to its low dimensionality: in total, it involves $3\times N$ features, where N is the number of topics. However, it is prone to overfitting, since the learned models depend on the absolute similarity value with the class graphs. For example, a decision tree that was trained to assign a document d_i to topic $\mathcal{T}_j$ if $CSim(G_{d_i}, G_{\mathcal{T}_j}) > 0.8$, will fail to correctly classify a document $d_k \in \mathcal{T}_j$ for which $CSim(G_{d_k}, G_{\mathcal{T}_j}) = 0.79$.

To overcome this shortcoming, we now introduce an alternative type of features, the *nominal* ones. Given N distinct topics and the numeric feature vector of an individual document, we compare the similarities of the same type in a pairwise manner. Each pair of numeric features $sim_{\mathcal{T}_i}$ and $sim_{\mathcal{T}_j}$ is replaced with a nominal value $dsim$ that corresponds to the class with the highest similarity value. More formally:

$$dsim(sim_{\mathcal{T}_i}, sim_{\mathcal{T}_j}) = \begin{cases} \mathcal{T}_i, & \text{if } sim_{\mathcal{T}_i} > sim_{\mathcal{T}_j} \\ equal, & \text{if } sim_{\mathcal{T}_i} = sim_{\mathcal{T}_j} \\ \mathcal{T}_j, & \text{if } sim_{\mathcal{T}_i} < sim_{\mathcal{T}_j}, \end{cases} \tag{1}$$

where sim denotes either $CSim$, $NVSim$ or $VSim$. In this way, we convert the original feature space of $3\times N$ numeric features into a new one consisting of $3\times N\times(N$-1)/2 nominal ones. Despite the higher dimensionality, the search space is reduced from $3\times N$ dimensions defined in the interval [0, 1] to $3 \times N \times (N-1)/2$ distinct labels. Regarding accuracy, the learned model is more robust to noise and to

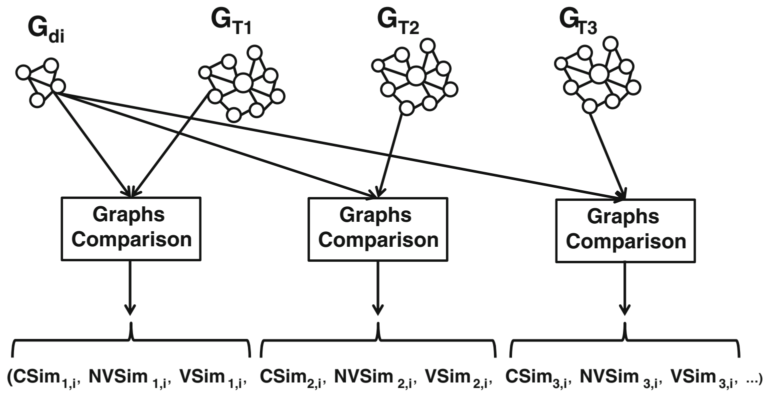
G_di
G_T1
G_T2
G_T3
Graphs Comparison
Graphs Comparison
Graphs Comparison
(CSim_1,i, NVSim_1,i, VSim_1,i, CSim_2,i, NVSim_2,i, VSim_2,i, CSim_3,i, NVSim_3,i, VSim_3,i, ...)

Figure 2 Extracting the feature vector from the n-gram graph model

overfitting, since the learned model is decoupled from the actual similarity values; instead, it depends exclusively on the document's relative similarity with all pairs of class graphs.

Note that the numeric and the nominal features are complementary and can be combined into *hybrid* classification schemes. We investigate the relative performance of the feature types in Section 5.4.

(ii) *Model granularity.* This term captures the type of substrings that is used to build the n-gram graphs. Similar to bag models, two are the valid options: character n-grams and token n-grams. The former are more robust to spelling mistakes and, thus, achieve higher accuracy in the context of noisy corpora. The advantage of the token n-grams is that they result in graphs of smaller size and order (i.e., fewer nodes and edges), thus involving a more efficient feature extraction process. We experimentally compare these two granularities in Section 5.5.

(iii) *Core size* (n). The higher the value of n, the more informative and distinctive the patterns captured by the n-gram graphs. This yields higher classification accuracy at the cost of lower time efficiency. In the following, we consider relatively small core sizes. For character n-grams, n is usually set to 2, 3 or 4, with the corresponding models called *character bigram graphs* (C2GG), *character trigram graphs* (C3GG) and *character four-gram graphs* (C4GG). For token n-grams, n is usually set to 1, 2 or 3 and the resulting models are called *token unigram graphs* (T1GG), *token bigram graphs* (T2GG) and *token trigram graphs* (T3GG), respectively. These core sizes correspond to those considered for bag models and lie within previously studied limits, which were theoretically and experimentally verified to involve a good balance between classification accuracy and time [17]. We empirically examine their relative performance in Section 5.5.

(iv) *Merge portion* (mp). This parameter expresses the portion of *labelled* documents that are merged into the class graph of each topic. The higher its value, the more diverse the patterns captured by the corresponding class graphs and the higher the resulting classification accuracy. However, high values lead to large class graphs that involve a high computational cost (i.e., low time efficiency) when estimating their similarity with document graphs for the extraction of feature vectors. In addition, the maximum value of the parameter results in including all training samples to the class graph, thus overfitting the data: all samples of a given class will be expected to have very high similarity to the corresponding class graph. In Section 5.3, we experimentally estimate the merge portion that yields the best balance between classification accuracy and time for each graph model and document type.

3.3 Qualitative analysis

Having outlined the bag and graph representation models, we now elaborate on their qualitative aspects, explaining how their performance is affected by the settings we are considering (i.e., the four challenges of Section 1).

For token n-grams, the most critical step is usually the identification of the distinct words in a collection of documents. The challenge is actually to detect and cluster together the different appearances of the same word; otherwise the resulting model suffers from low efficiency (due to an excessively large feature space) and low effectiveness (due to missed patterns) [14]. Three are the main obstacles to this effort:

- Noise in the form of spelling mistakes (i.e., challenge Ch2).
- Synonymy, i.e., the phenomenon where different words that have identical meanings. For instance, the words “buy” and “purchase”.
- Polysemy, i.e., the phenomenon where identical words that have different meanings. For instance, the word “left” can refer either to the past tense of leave or to the opposite of right.

To tackle these issues, pre-processing methods are usually employed with the aim of grouping together different manifestations of the same word or meaning [29]. *Stemming* reduces the inflected or derived words to their root form, usually by removing their suffix (e.g., it removes the plural “s” from the nouns in English). *Lemmatization* improves on this process by taking into account the context of a word – or even grammar information – in order to match it to a lemma. *Part-of-speech tagging* infers the lexical or grammatical category of a specific word inside a phrase or sentence. The drawback of these techniques is that they require language-dependent knowledge, thus having limited effectiveness in multilingual settings (i.e., challenge Ch1).

In contrast, the character n-gram models convey a language-neutral functionality that is inherently robust to noise, especially with respect to spelling mistakes. Thus, they help to deal with challenges Ch1 and Ch2, respectively. They also allow for fuzzy and substring matching, which constitute functionalities of high importance in open domains, like the content of Social Media (i.e., challenge Ch4). Similar to token n-grams, though, they suffer from the *curse of dimensionality*: the number of features that they entail is usually very high – depending, of course, on the size and the vocabulary of the given corpus. For the same corpus, the feature space of character n-grams is typically larger than that of the token ones, increasing its dimensionality with the increase of n. The reason is that larger core sizes result in a higher number of possible character combinations. This situation is particularly aggravated in the context of a highly diverse vocabulary: the more heterogeneous a document collection – either with respect to the languages it comprises (i.e., challenge Ch1) or the vocabulary variations introduced by its authors (i.e., challenge Ch4) – the higher is the number of features that the bag models take into account. A common, language-agnostic practice for restricting the impact of dimensionality is to set a threshold on the minimum document frequency of the n-grams that are considered as features. This practice, however, is a mere application-dependent heuristic that may have a negative impact on accuracy.

Regarding the graph models, the token n-gram graphs have the same limitations as the token n-grams, while the character n-gram graphs have the same advantages as the character n-grams. That is, they involve a noise-tolerant functionality for dealing with challenge Ch2 and a language-agnostic functionality for dealing with challenge Ch1. The main difference between bag and graph models is that the former disregard the order of n-grams appearance in the original text, thus losing valuable information. As a result, documents or topics with common n-grams that appear in completely different sequences, end up having highly similar bag representations. The graph models overcome this problem through their weighted edges, which capture contextual information in the form of co-occurring n-grams. Their edges actually enable them to detect more accurate and more reliable patterns even in the context of sparse documents, thus increasing the performance over challenge Ch3. Therefore, each graph model is expected to yield higher effectiveness than its bag counterpart.

Another advantage of graph models is that they do not suffer from the curse of dimensionality; they involve a limited feature space, the size of which depends exclusively on the number of considered classes. Thus, their search space is independent of the challenges Ch2 and Ch4, involving a significantly lower computational cost than bag models for the training of classification algorithms as well as the use of trained models. Their only drawback with respect to efficiency is the time required for the construction of class graphs and the computation of graph similarities. The time complexity of these processes depends on the size and the type of the input document collection as well as on the internal configuration of graph models. In Section 5.3, we experimentally fine-tune them with respect to merge portion in order to enhance their time efficiency at a limited cost in classification accuracy.

4 Web document types

In this section, we focus on the traits of Web documents that affect the classification accuracy and time of document representation models. We identify the most critical parameters for the performance of TC and use them to distinguish Web documents into three main types.[6] In this effort, we exclusively consider *endogenous parameters*, which can be directly derived from the textual content of Web documents (as mentioned above, exogenous metadata are beyond the scope of this work). We propose the following criteria as the most influential for the performance of topic classifiers:

(i) The *document size* (DS) expresses the average length of documents in terms of the number of characters they comprise. Given a document collection $\mathcal{D}$, it is formally defined as:

$$DS(\mathcal{D}) = \frac{\sum_{d_i \in \mathcal{D}} |d_i|}{|\mathcal{D}|},$$

where $|d_i|$ stands for the individual length of document $d_i \in \mathcal{D}$ and $|\mathcal{D}|$ denotes the size of the corpus. DS is directly related to challenge Ch3, since the shorter a document, the more sparse the information it contains. It also pertains to the computational cost of the classification process: the larger a document, the higher the computational

[6]It is worth stressing that these three types do not correspond to document genres; instead, the aim is to explain the difference in the quality of Web documents and the resulting impact on TC.

cost, due to the higher number of possible n-grams that can be used as features and nodes by the bag and graph models, respectively.

(ii) The *vocabulary size* (VS) denotes the diversity of the words and phrases that are employed in a document collection. Higher diversity (i.e., higher VS) corresponds to a higher number of n-grams that can be possibly used as features and nodes by the bag and graph models, respectively. Given a document collection $\mathcal{D}$, VS is defined as the ratio of distinct tokens over their total number of occurrences:

$$VS(\mathcal{D}) = \frac{|\bigcup_{d_i \in \mathcal{D}} tokens(d_i)|}{\sum_{d_i \in \mathcal{D}} tokensNumber(d_i)},$$

where $tokens(d_i)$ stands for the set of tokens appearing in document $d_i \in \mathcal{D}$, and $tokensNumber(d_i)$ denotes the total number of tokens in d_i. VS increases with multilinguality (i.e., challenge Ch1) and with the evolving, non-standard expressions used in Social Media content (i.e., challenge Ch4). In curated Web documents, though, the vocabulary exhibits lower levels of diversity (e.g., news articles).

(iii) *Noise* is directly related to challenge Ch2, reflecting the portion of spelling mistakes and of grammatically, syntactically and semantically incorrect phrases in a document collection. Such errors distort the actual meaning of a phrase or sentence and hinder the detection of token and character n-gram patterns. They are rather difficult to be directly measured. Instead, we assess them indirectly, through the frequency distribution $\mathbf{f}$ of the tokens appearing in the given document collection: noisy corpora abound in erroneous tokens with a single or few occurrences (due to unrepeated errors), while clean corpora are dominated by tokens that occur multiple times. This frequency distribution is summarized through three statistical metrics: the 1^{st} quartile ($Q_1(\mathbf{f})$), the median or 2^{nd} quartile ($Q_2(\mathbf{f})$) and the 3^{rd} quartile ($Q_3(\mathbf{f})$). High levels of noise correspond to similar, low values across all metrics; for instance, $Q_3(\mathbf{f}) = 1$ means that at least 75 % of all tokens appear only once. In contrast, low levels of noise yield a large discrepancy in the values of these metrics; the higher the difference between the median and the other two quartiles, the lower the levels of noise and the higher the reliability of the content-based features.

(iv) *Special Notation* denotes the extent to which a document contains non-verbal expressions. These include HTML code, links to Web pages and multimedia content (URLs) as well as pointers to other users (e.g., the @*username* notation used in Twitter). This kind of special notation is typically exploited by contextual representation models, which try to enhance the effectiveness of TC by incorporating exogenous meta-data (e.g., [26]). As a content-based feature, though, special notation adds noise and, thus, it is relevant to challenge Ch2. To measure its effect, we simply estimate the portion of tokens that correspond to non-verbal expressions.

We argue that these four criteria capture the main characteristics of the major types of contemporary Web documents: static Web documents, discussion fora and blogs as well as Social Media messages. In more detail, we distinguish the following three categories of Web documents:

– *Curated* documents are large and contain almost exclusively pure text (i.e., absence of special notation), written in a specific language. They also use a standard, formal

vocabulary that typically lacks grammatical and syntactic errors.
- *Semi-curated* documents are shorter in size and contain special notation usually in the form of hyperlinks. They have higher levels of noise, a more diverse vocabulary and their content may be multilingual.
- *Raw* documents are rather telegraphic, noisy and rich in various types of special notation.

We elaborate on each Web document type in the following, analysing its implications to the process of topic classification.

4.1 Curated documents

This type comprises such documents as news articles, scientific publications, books and literary works. The text is adequately long to pose well-described questions, to provide argumentation or to cover a topic. Therefore, challenge Ch3 (i.e., sparseness) is not an issue. The same applies to challenge Ch2, as well: the content has been edited or peer-reviewed and the writing is mostly correct. Its language is eloquent and the text itself is focused and clear, without any neologisms and non-standard vocabulary. Thus, it does not suffer from challenge Ch4, involving controllable levels of diversity. In addition, the content is rarely multilingual, but even when multiple languages are used (i.e., challenge Ch1), they are usually known a priori. As a result, their tokens can be easily transformed into reliable features through lemmatization and stemming.

In this context, token n-grams are expected to offer a better balance between classification accuracy and time than character n-grams: their lower dimensionality ensures higher efficiency, while the negligible levels of noise make them equally accurate. Similarly, token n-gram graphs involve smaller and, thus, more efficient graphs than character n-gram graphs. The lower dimensionality of graph models allows for a more efficient learning procedure, but the large size of the curated documents increases the computational cost for the creation of document and class graphs and the execution of the graph comparisons for the extraction of feature vectors.

4.2 Semi-curated documents

Documents of this type come in the form of forum posts, text in wikis, e-mails and personal blog posts. They are not necessarily written in a single, known language (i.e., challenge Ch1) and their content is minimally edited by the author, with spelling mistakes and wrong sentences being relatively common (i.e., challenge Ch2). Usually, they comprise few paragraphs (i.e., challenge Ch3), which are, however, long enough to analyse personal thoughts or to act as written dialogue parts. Neologisms, informal language and special notation are frequently used (i.e., challenge Ch4). On the whole, the semi-curated documents involve to some degree all challenges of Section 1.

For token n-grams, the medium document and vocabulary size is expected to lead to a middle-sized feature space. Similarly, token n-gram graphs yield medium-sized document and class graphs that require affordable computational cost. Most importantly, though, the presence of noise is expected to have a significant impact on the accuracy of token-based models. Instead, the character-based ones are able to achieve higher accuracy, due to their inherent robustness to noise. They suffer, though, from lower efficiency than the corresponding token models, due to the larger feature space and the more complex graphs they entail.

4.3 Raw documents

This type refers to such documents as Facebook[7] status updates, YouTube[8] comments, messages in Twitter (also termed *tweets*) and short posts in any Web 2.0 platform. A basic trait of these documents is that they are meant to be self-contained, conveying their message through a text of minimal size (i.e., challenge Ch3). For instance, the messages are often meant to be the answer to questions like "What is new?", "What are you thinking?", "What is happening?". They can also comprise brief comments that simply convey an opinion or sentiment. Their authors typically use the full range of internet neologisms, abbreviations, emoticons and similar language constructs (i.e., challenge Ch4). The quality of the text is usually of minimal interest, since an erroneous or even incomprehensible part can be simply explained with a new message (i.e., challenge Ch2). In addition, it is not rare for users to employ a mixture of languages, even in a single message (i.e., challenge Ch1). They also make frequent use of high levels of geographic lexical variations (i.e., challenge Ch4). For instance, Twitter users from northern California write "koo" instead of cool, while the same word in southern California is mentioned as "coo" [11]. In summary, raw documents contain short, unedited, and noisy texts and are abundant in special notations that may be essential to understand their meaning.

The high levels of noise in combination with sparseness pose a significant barrier to the accuracy of token n-grams. Token n-gram graphs overcome sparseness to some extent, thus outperforming their bag counterparts. Their classification accuracy, though, is also limited by noise. In contrast, character-based models are expected to perform significantly better, due to their inherent robustness to noise. Among them, character n-gram graphs offer higher accuracy than character n-grams, since they are better equipped to deal with sparseness. In addition, the limited size of raw documents favours the efficiency of the more expressive (character) graph models, minimizing the computational cost for the creation and comparison of graphs.

4.4 Discussion

The characterization of the three document types with respect to the above-mentioned criteria is summarized in Table 1. We can argue that the curated and the raw documents define the two extremes of Web document quality with respect to morphology. The former involves large texts, where spelling mistakes and non-standard expressions are the exception, while the latter entails short, noisy texts with non-standard, slang expressions and a considerable proportion of special notation. Semi-curated documents lie in the middle of these two extremes, involving medium-sized texts and medium levels of noise, of non-standard expressions and of special notation. We provide experimental evidence for these patterns in Section 5.2.

5 Experimental evaluation

The goal of this section is threefold: (i) To provide quantitative evidence for our characterization of Web documents. (ii) To fine-tune the "merge portion" and the "feature type"

[7]http://www.facebook.com

[8]http://www.youtube.com

Table 1 Content-based, qualitative taxonomy of Web documents

	Document size	Vocabulary size	Special notation	Noise
Curated documents	High	Low	Negligible	Negligible
Semi-curated documents	Medium	Medium	Low	Low
Raw documents	Low	High	High	High

of graph models so as to achieve the best balance between effectiveness and efficiency for each document type. (iii) To thoroughly compare the established bag models with the graph ones across all document types with respect to the common parameters of "core size" and "model granularity".

We begin our experimental analysis with the presentation of the datasets and the evaluation metrics in Section 5.1. We then perform a statistical analysis that provides grounds for our document types in Section 5.2. We continue by configuring the merge portion and the feature type of graph models in Sections 5.3 and 5.4, respectively. We analytically compare bag and graph models in Section 5.5 and we conclude our experimental study with a summarization of its main outcomes in Section 5.6.

5.1 Setup

All experiments were fully implemented in Java, version 1.6, and were performed on a server with Intel i7-3820 (3.60GHz) and 32GB of RAM memory, running Debian 7.0. The functionality of n-gram graphs was provided by the open source library of JInsect.[9]

To derive the performance of the representation models, we applied them to two established classification algorithms that are typically used for TC in conjunction with the bag models: Naive Bayes Multinomial (NBM) and Support Vector Machines (SVM) [29, 44]. The former classifies instances based on the conditional probabilities of their feature values, while the latter uses optimization techniques to identify the maximum margin decision hyperplane. For their implementation, we used the open source library of Weka,[10] version 3.6 [19]. For NBM, this library provides a parameter-free implementation, whereas for SVM, we employed the default configuration of Weka, without fine-tuning any of its parameters: the complexity constant was set to 1 and a linear polynomial was used as the kernel function.

5.1.1 Datasets

To evaluate all representation models in real settings, we considered three large-scale, real-world datasets – one for each type of Web document. Their technical characteristics are presented in Table 2. We briefly describe them in the following paragraphs.

Curated documents As representative for this type of document, we selected the Reuters RCV2 corpus,[11] which has been widely used in the literature [1, 7]. It comprises a multilingual collection of news articles, published in the time period between August 1996 and

[9]http://sourceforge.net/projects/jinsect

[10]http://www.cs.waikato.ac.nz/ml/weka

[11]http://trec.nist.gov/data/reuters/reuters.html

Table 2 Class distributions of the datasets employed in our experimental study

D_{reuters}		D_{blogs}		D_{twitter}	
Class	Distribution	Class	Distribution	Class	Distribution
ECAT	13,768 (08.00 %)	Current affairs	3,288 (04.51 %)	#quote	8,215 (02.50 %)
MCAT	41,523 (24.13 %)	Entertainment	3,751 (05.15 %)	#fact	13,144 (04.00 %)
CCAT	45,382 (26.37 %)	Blog	3,825 (05.25 %)	#followfriday	18,322 (05.57 %)
GCAT	71,442 (41.50 %)	Work	4,095 (05.62 %)	#news	21,236 (06.46 %)
		Life	4,631 (06.35 %)	#musicmonday	25,520 (07.76 %)
		Personal	5,003 (06.86 %)	#iranelection	26,406 (08.03 %)
		Politics	6,738 (09.24 %)	#tcot	30,012 (09.13 %)
		Music	8,295 (11.38 %)	#ff	35,458 (10.78 %)
		News	12,970 (17.79 %)	#jobs	71,584 (21.77 %)
		Votes	20,320 (27.87 %)	#fb	78,898 (24.00 %)
Total	172,115 (100.0 %)	Total	72,916 (100.0 %)	Total	328,795 (100.0 %)

August 1997. In total, it contains over 480,000 articles that are written in 13 different languages. For our analysis, we considered a subset of this collection, comprising 172,115 articles that are written in four languages: German, Spanish, Italian, and French. The news articles of RCV2 are categorized along a class hierarchy of 104 overlapping topics. In our experiments, we considered only the top four categories, because they are non-overlapping and, thus, are compatible with single-label TC. The selected topics along with the class distributions are depicted in Table 2. This data collection is denoted by D_{reuters} in the rest of the paper.

Semi-curated documents For this type of document, we selected the collection of blog posts that was published in the context of the 3^{rd} workshop on the Weblogging Ecosystem in 2006.[12] It contains around 10 million documents, stemming from approximately 1 million different weblogs. They were posted on-line in the time period between July 7, 2005 and July 24, 2005. For our analysis, we considered the 10 largest categories, removing those documents that belong to more than one of them, since we examine single-label TC. This procedure yielded 72,916 blog posts, the class distribution of which is presented in Table 2. Unfortunately, there is no information about the languages they are written in. In the following, this dataset is indicated as D_{blogs}.

Raw documents This type of document is represented in our analysis by Twitter posts. We extracted our dataset from that of [45], which comprises 467 million tweets that have been posted by around 20 million users in the time interval between June, 2009 and December, 2009. To derive the topic categorization of the tweets, we relied on their *hashtags*.[13] Around 49 million of the tweets are marked with at least one hashtag. As with D_{blogs}, we considered only those documents that exclusively belong to one of the 10 largest topics (i.e., hashtags). We also excluded all the retweets, because they lack any original information, being mere

[12] http://www.blogpulse.com/www2006-workshop/datashare-instructions.txt

[13] A hashtag in Twitter consists of the symbol #, followed by a series of concatenated words and/or alphanumerics (e.g., #worldcup2014).

reproductions of other tweets. This procedure yielded around 3.5 million tweets. To restrict the dataset to a manageable size, we randomly selected 1/10 of these documents such that the relative sizes of the topics were maintained. Finally, we removed all hashtags from the remaining documents, since they contain category information [15, 26]. The resulting collection – represented by $\text{D}_{\text{twitter}}$ in the following – comprises more than 300 thousand tweets. Their class distribution is presented in Table 2. Again, there is no information about the languages the documents are written in.

5.1.2 Metrics

Our experimental evaluation emphasizes the trade-off between effectiveness and efficiency for each representation model. To measure the former aspect, we employ the metric of *classification accuracy* (α). Given a corpus of documents $\mathcal{D}$, the accuracy of single-label TC is defined as:

$$\alpha = \frac{|\mathcal{D}| - |E|}{|\mathcal{D}|} \cdot 100\ \%,$$

where $|E|$ is the number of classification errors (i.e., documents that were assigned to a false class – cf. Problem 1) and $|\mathcal{D}|$ denotes the number of documents in the given corpus. To get a robust estimate of this metric in each experiment, we applied the 10-fold cross-validation scheme to all classification settings we consider in the following. We use the accuracy of these 10 folds in order to examine whether the performance of two models differs significantly for a specific dataset and classification algorithm. In more detail, we apply the two-tailed Wilcoxon signed-ranks test to the accuracy of the 10 folds and consider the difference between two models to be *statistically significant* only if $p < 0.05$.

To measure the time efficiency of a representation model, we use the metric of *classification time* (t_c). It estimates the time required for extracting the feature vector from an individual unlabelled document and for applying the trained classification model to it. In our experimental study, the classification time values correspond to the average value over the 10 folds (i.e., iterations). Note that we disregard the temporal requirement of the training procedure, because it is executed only once and off-line. Instead, the classification time denotes the cost of repeatedly using the learned model on-line.

Before elaborating on the experimental classification performance, we delve deeper in the different document types and how these manifest on quantifiable criteria.

5.2 Analysis of document types

In this section, we assess the four content-based criteria that were used for our characterization of Web documents. Quantitative evidence for the first three is provided in Table 3, while special notation is examined separately in Table 4.

Starting with document size (DS), we can see that it exhibits a large variation across the three datasets: on average, the news articles of $\text{D}_{\text{reuters}}$ contain 1,200 characters or 180 tokens, while the raw documents of $\text{D}_{\text{twitter}}$ are very sparse, consisting of just 130 characters or 21 tokens. D_{blogs} lies closer to $\text{D}_{\text{reuters}}$, comprising documents with 1,100 characters or 160 tokens, on average.

Regarding vocabulary size (VS), we observe that it is limited for the curated documents of $\text{D}_{\text{reuters}}$ and varies the most for the raw documents of $\text{D}_{\text{twitter}}$. The diversity of the latter is actually triple than that of the former: on average, every 100 tokens in $\text{D}_{\text{twitter}}$ contain 11 distinct terms, compared to less than 4 in $\text{D}_{\text{reuters}}$. D_{blogs} lies right in the middle of these two extremes, with 7 unique terms for every 100 tokens, on average.

Table 3 Quantifying the content-based criteria that determine the type of Web documents. DS and VS denote the document and the vocabulary size, respectively, whereas $Q_x(\mathbf{f})$, $x \in \{1, 2, 3\}$, represents the quartiles of the frequency distribution of tokens

		$D_{reuters}$	D_{blogs}	$D_{twitter}$
DS	Total characters	2.07×10^8	8.02×10^7	4.29×10^7
	Document size	1,206	1,100	130
	Tokens per document	180	160	21
VS	Total tokens	3.09×10^7	1.17×10^7	6.97×10^6
	Distinct Tokens	1.14×10^6	8.23×10^5	7.69×10^5
	Vocabulary size	3.70×10^{-2}	7.04×10^{-2}	11.02×10^{-2}
Noise	$Q_1(\mathbf{f})$	1	1	1
	$Q_2(\mathbf{f})$	1	1	1
	$Q_3(\mathbf{f})$	3	2	1

Considerable variation is also observed with respect to noise, as denoted by the 3^{rd} quartile of document frequency. $Q_3(\mathbf{f})$ takes the largest value for $D_{reuters}$, suggesting that its individual tokens appear more frequently and, thus, are less likely to contain errors, such as spelling mistakes. In contrast, the lowest value (i.e., 1) corresponds to $D_{twitter}$, implying that at least 75 % of all of its tokens appear just once. In fact, 76 % of all tokens in $D_{twitter}$ correspond to a single occurrence, compared to just 55 % in $D_{reuters}$. D_{blogs} lies in between these two extremes ($Q_3(\mathbf{f})$=2), but closer to $D_{twitter}$, with 70 % of its tokens appearing just once.

Regarding special notation, we can see in Table 4 that it is totally absent from the curated documents of $D_{reuters}$. For D_{blogs}, it is restricted to URLs, which merely account for 0.01 % of all tokens. In the case of $D_{twitter}$, the relative amount of special notations is significantly higher, with more than 10 % of all words corresponding to "non-verbal tokens": 3.6 % of all tokens refer to some Twitter user (i.e., mentions), 2.3 % are URLs and 4.8 % designate the topic(s) of the tweet (i.e., hashtags). It is also interesting to note that its regular words have an average length of 4.8 characters, thus being smaller than those of $D_{reuters}$ by a whole character. This should be expected, because raw documents contain abbreviations and neologisms, which are typically shorter than the original words (e.g., "gr8" instead of "great"). This phenomenon does not appear in D_{blogs}, as the length of its regular tokens coincides with that of $D_{reuters}$.

On the whole, these traits validate quantitatively our content-based criteria as well as the types of Web document that we defined in Section 4.

Table 4 Analysis of special notation for all document types

	$D_{reuters}$		D_{blogs}		$D_{twitter}$	
	Length	Occurrences	Length	Occurrences	Length	Occurrences
Mention	–	–	–	–	11.5	2.54×10^5 (03.64 %)
HashTag	–	–	–	–	5.9	3.31×10^5 (04.75 %)
URL	–	–	60.7	1.21×10^3 (00.01 %)	22.5	1.59×10^5 (02.28 %)
Regular Word	5.7	3.09×10^7	5.7	1.17×10^7 (99.99 %)	4.8	6.23×10^6 (89.33 %)

5.3 Merge portion configuration

This section investigates how fast the n-gram graphs converge to an accurate model, an aspect that is principally determined by the merge portion. Remember that mp specifies what portion of each topic's labelled instances participate in the creation of the corresponding class graph. As explained in Section 3.2, there is a fundamental trade-off regarding mp: the higher its value, the higher the classification accuracy and the lower the efficiency of the representation model. Therefore, our analysis aims at estimating the merge portion that yields the best balance between classification accuracy and time for each representation model and document type.

We exclusively consider numerical features in this analysis, since the nominal and the hybrid ones yield similar patterns. We applied all graph models to the three datasets of our study and trained NBM over the resulting numeric features (SVM exhibited similar behavior and is omitted for brevity). For each combination of graph model and feature type, we derived its performance by incrementing the merge portion from 0.1 to 1.0 with a step of 0.1. Note that instead of $D_{reuters}$, we employed a random selection of half its documents that retains the relative size of the classes. This sample, which is denoted by $D'_{reuters}$, was preferred over the entire $D_{reuters}$ in order to reduce the originally massive dataset to a moderate size that facilitates our thorough experimental analysis.

The resulting performances are presented in Figure 3a to f. The diagrams on the left column indicate the learning curves of the representation models, depicting the evolution of their accuracy α with respect to mp. The diagrams on the right column indicate the evolution of their efficiency in terms of the classification time, t_c. The first row of figures corresponds to $D_{reuters'}$, the second to D_{blogs} and the third to $D_{twitter}$.

Starting with accuracy, we observe that α increases modestly with the increase of the merge portion across all datasets and graph models, regardless of their core size and granularity. The increase is steeper for smaller merge portions, but becomes rather insignificant after $mp = 0.5$ in most of the cases. The only exception to this pattern is C2GG: its effectiveness remains practically stable over D_{blogs} and $D_{twitter}$ and drops steadily for higher merge portions over $D_{reuters'}$.

Regarding the classification time, we observe that t_c increases monotonically (almost linearly) with merge portion. The reason is that it is insensitive to the construction of the document graph and the use of the trained model, which together account for less than 1 % of the overall classification time. Instead, t_c is dominated by the time required for the comparison between document and class graphs, with the size of the latter increasing linearly with mp, i.e., the number of documents merged into them [17]. The maximum value of classification time is higher than the lowest one by 2 to 20 times across all datasets and representation models, with larger core sizes yielding higher differences.

On the whole, we can conclude that larger mp values lead to a significantly higher computational cost as expressed through t_c, while yielding a moderate increase in classification accuracy, α. Therefore, lower values of merge portion achieve a better balance between effectiveness and efficiency in most of the cases. We can quantify this balance for each value of mp through its *utility ratio*, $u(mp)$, which expresses the ratio between the gain in efficiency and the cost in effectiveness that mp conveys. More formally, this ratio is defined as:

$$u(mp) = \frac{gain(mp)}{cost(mp)} = \frac{\frac{t_c^{max} - t_c(mp)}{t_c^{max}}}{\frac{\alpha^{max} - \alpha(mp)}{\alpha^{max}} + 1},$$

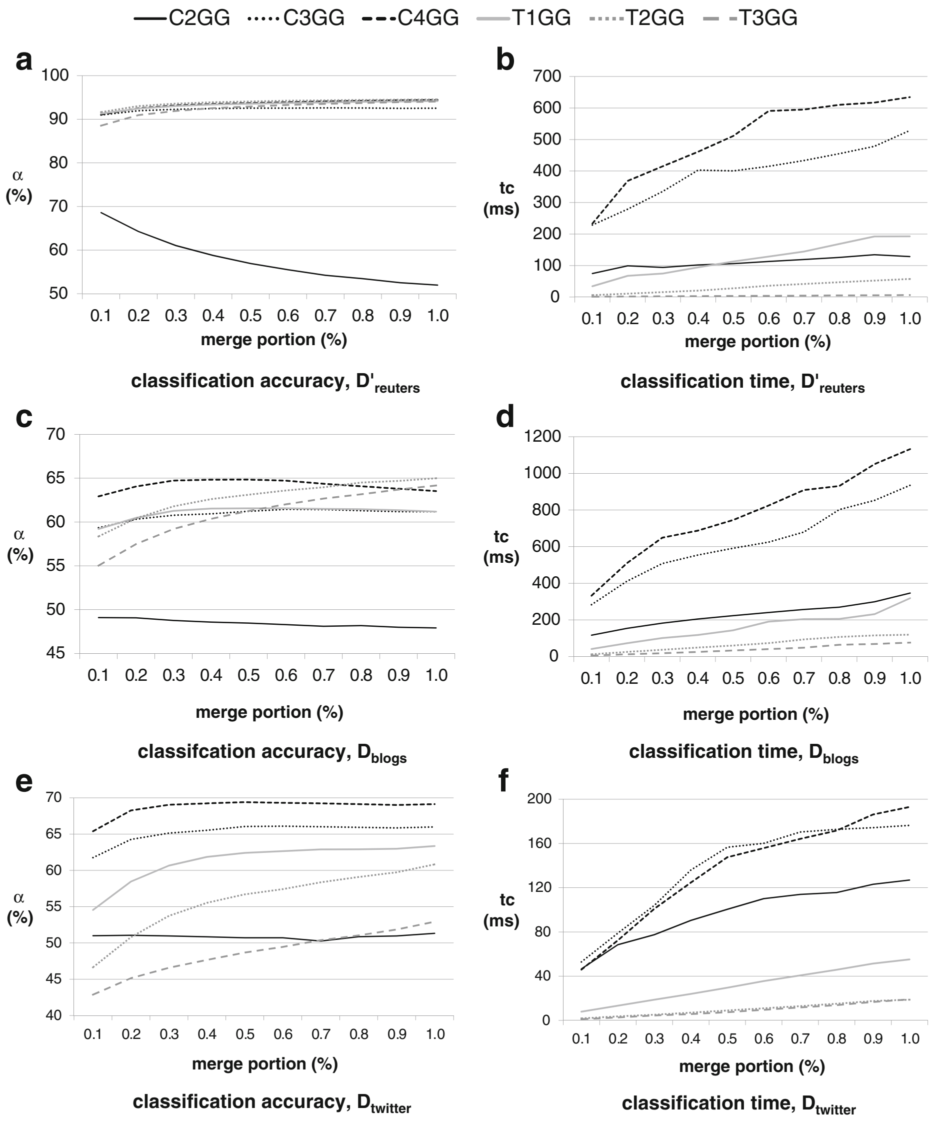

Figure 3 The effect of merge portion on the evolution of accuracy α in (**a**), (**c**) and (**e**) and of classification time t_c in (**b**), (**d**) and (**f**) over $D_{reuters'}$, D_{blogs} and $D_{twitter}$, respectively

where α^{max} and t_c^{max} correspond to the highest classification accuracy and time, respectively, across all the merge portions that we consider for a specific representation model and dataset. In other words, the numerator denotes the relative gain in classification time, while the denominator stands for the relative cost in classification accuracy with respect to the maximum values of these metrics. Note that the denominator is incremented by 1 so as to avoid infinite utility values for the merge portions with the maximum accuracy.

Based on this measure, we can specify as *optimal merge portion* for a specific graph model and dataset the one maximizing its utility ratio. The configurations resulting from this rule are depicted in Table 5. We observe that in 12 out of 18 cases, mp is lower than 0.5. This means that less than half the training instances of each topic are typically needed for building a comprehensive class graph. This is particularly true for the character graph models, which consistently yield the lowest merge portions across all corpora. Note, though, that for both character and token models, the larger the core size, the larger the optimal mp – regardless of the document type. This practically means that the smaller the core size n, the fewer the possible combinations of n-grams and the less documents suffice for constructing representative class graphs that capture most distinguishing patterns. The best example for this is C2GG, whose optimal mp takes the smallest possible value across all document types.

In all the following analyses, we employ the mp configuration of Table 5, regardless of the underlying classification algorithm and feature type, considering that it will provide near-optimal performance in any similar setting.

5.4 Feature type configuration

The goal of this section is to identify the type of features that yields the best performance for every graph model and document type. To this end, we applied NBM and SVM to all feature types and models across all datasets using the merge portion configuration of Table 5. The outcomes indicate that there is no practical difference in the classification time required by the three feature types for a specific model and dataset. As explained above, the reason is that t_c is dominated by the comparisons between the document and the class graphs, a procedure that is involved in the extraction of any type of feature. Thus, we exclusively consider the classification accuracy in this analysis and report the temporal requirements of the graph models in the next section.

The actual accuracy α across all datasets and models is presented in Figure 4a to f. The first row corresponds to $D_{reuters}$ (the entire dataset), the second to D_{blogs} and the third to $D_{twitter}$. The figures on the right column present the performance of SVM, while those on the left correspond to NBM (the nominal features were applied to Naive Bayes instead of NBM, because it leverages them to a greater extent). Note that we use a different scale for every dataset, which is consistent for both classification algorithms.

Starting with Figure 4a and b, we observe that all feature types achieve an accuracy between 92 % and 96 % in most cases – regardless of the representation model and the classification algorithm. This indicates that the comprehensive, noise-free content of curated documents is equally well represented by any feature type. It also explains why the simple learned models of NBM achieve equivalent performance with the more complicated ones of SVM. Yet, we can identify the following pattern: the numerical and the hybrid features exhibit a practically identical accuracy and take a minor, but statistically significant lead over the nominal ones. There are two exceptions to this pattern. First, the nominal features outperform the numerical and the hybrid ones by more than 25 % and 1.5 %, respectively,

Table 5 The merge portions that yield the best balance between classification accuracy and time for each dataset and graph model

	C2GG	C3GG	C4GG	T1GG	T2GG	T3GG
$D_{reuters'}$	0.1	0.2	0.3	0.3	0.4	0.4
D_{blogs}	0.1	0.6	0.4	0.4	0.5	0.7
$D_{twitter}$	0.1	0.3	0.3	0.5	0.5	0.5

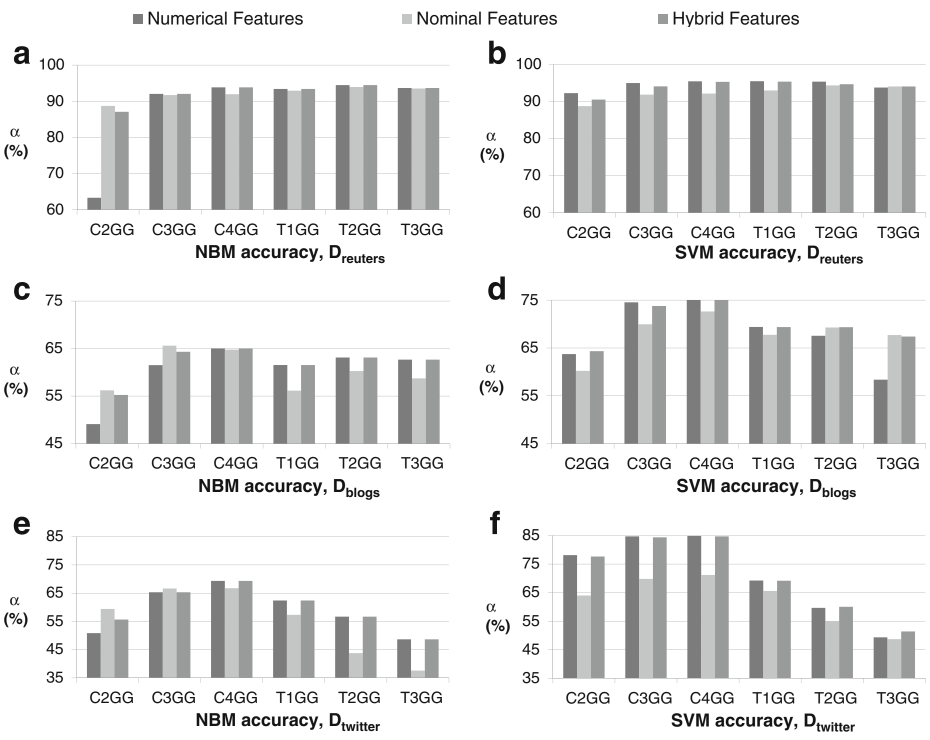

Figure 4 The effect of feature type on the classification accuracy α with respect to NBM in (**a**), (**c**) and (**e**) and to SVM in (**b**), (**d**) and (**f**) over $D_{reuters}$, D_{blogs} and $D_{twitter}$, respectively

for C2GG in combination with NBM. Second, the differences between the three feature types are insignificant in the case of T3GG.

Figure 4c and d demonstrate the performance of the feature types over the semi-curated content of D_{blogs} in combination with NBM and SVM, respectively. Again, we observe the predominance of numerical and hybrid features, which coincide and surpass the nominal ones for both algorithms across most representation models. For this dataset, though, there are more exceptions to the prevalent pattern. In fact, the nominal features outperform the numerical ones for C2GG and C3GG in conjunction with NBM and for T3GG in conjunction with SVM. In all these cases, the difference in accuracy is statistically significant, while the hybrid features are very close to the nominal ones. Note also that for C4GG, all feature types achieve practically identical accuracy when applied to NBM.

Regarding the raw content of $D_{twitter}$, Figure 4e and f present the performance of all feature types in combination with NBM and SVM, respectively. Again, the numerical and the hybrid features coincide in most cases, but now their prevalence is more intense, increasing their distance from the nominal ones. Similar to D_{blogs}, this pattern is reversed for C2GG and C3GG in conjunction with NBM. Different from $D_{reuters}$ and D_{blogs}, there is a setting where the hybrid features outperform the other types to a statistically significant extent ($p < 0.005$ for two-tailed Wilcoxon signed-ranks test), namely the combination of T3GG with SVM.

On the whole, we can argue that the nominal features make a difference only for weak representation models that are applied to noisy documents (i.e., semi-curated or raw).[14] In all other cases, the numerical features outperform them to a significant extent. Most importantly, though, the hybrid features are consistent in coinciding with the most accurate feature type under any settings. They actually exhibit the most robust behaviour with respect to α, while having identical classification times, as explained below. For this reason, we can conclude that they constitute the best feature type for all graph models – regardless of the document type and the classification algorithm.

5.5 Bag vs. graph models

We now compare the performance of bag and graph models with respect to their relative effectiveness and efficiency. In this analysis, we pay special attention to the effect of the two parameters that are shared by both model types: the core size, n, and the model granularity (i.e., character- or token-based models).

As before, we applied all models to NBM and SVM. For the graph models, we exclusively consider the hybrid features. For the bag ones, we use term frequency and TF-IDF as feature weights for the character- and the token-based models, respectively. To restrict their feature space to manageable sizes, we disregard the features that appear in a limited portion of the training set. We actually set the limit on minimum document frequency to 1 % of all labelled instances for $D_{reuters}$ and D_{blogs} and to 0.2 % for $D_{twitter}$. In this way, each model has a similar number of features across all datasets. The actual dimensionalities per model and dataset are presented in Table 6. Apparently, the bag models involve one or two orders of magnitude more features than their graph counterparts. To scale SVM to their large feature spaces, we employed the LibLinear optimization technique [12] through its Weka API. Given that LibLinear uses linear kernels for training the SVM, it is directly comparable with the default configuration of SVM in Weka (i.e., SMO), which was applied to graph models and incorporates linear kernels, too.

The outcomes with respect to classification accuracy over NBM and SVM are presented in Figures 5 to 7. The corresponding classification times are reported in Table 7. Similar to the graph models, the time efficiency of the bag ones is dominated by the extraction cost of the feature vector, as the use of the trained model merely accounts for a tiny portion of t_c (less than 1 %); for this reason, we present a single estimation for the classification time of each model and dataset, which amounts to the average across all folds of the two classifiers. In all cases, Cn collectively denotes the bag and graph character-based models with core size n, while Tn stands for the bag and graph token-based models with core size n.

In the following, we examine the relative performance of bag and graph models independently for each document type.

Curated documents Figure 5a and b present the accuracy of all models over $D_{reuters}$ in conjunction with NBM and SVM, respectively. We observe that the classification accuracy of character-based bag models increases in proportion to the core size and vice versa for the token-based ones. For both granularities, larger core sizes convey higher classification time. As a result, T1G consistently outperforms the other token-based models with respect to both

[14]The nominal features are also useful for powerful classification algorithms that are inherently crafted for this kind of evidence, such as C4.5. However, preliminary experiments demonstrated that such algorithms do not scale well to the large search space of bag models. Hence, we do not consider them in our analysis.

Table 6 Dimensionality of the feature space of every model over each dataset

		C2	C3	C4	T1	T2	T3
$D_{reuters}$	Bag models	1,989	8,336	15,454	1,771	1,105	976
	Graph models	31	31	31	31	31	31
D_{blogs}	Bag models	1,822	6,203	12,051	1,495	1,122	950
	Graph models	166	166	166	166	166	166
$D_{twitter}$	Bag models	1,965	6,128	9,907	1,108	931	786
	Graph models	166	166	166	166	166	166

metrics. T1G actually excels in effectiveness and in efficiency across all bag models, when using NBM. This pattern stems from the absence of errors and non-standard vocabulary in the content of curated documents (challenges Ch2 and Ch4) and can be extended to the average performance of character- and token-based models over NBM: the latter outperform the former in both evaluation metrics. Yet, SVM leverages character bag models to such an extent that they significantly exceed the accuracy of the token-based ones in almost all cases.

Similar patterns are exhibited by the graph models. In general, higher core sizes increase the accuracy and the classification time for the character-based models and decrease them for the token-based ones; t_c is actually reduced by a whole order of magnitude when moving from T1GG to T3GG. In the case of NBM, though, T2GG achieves the highest accuracy among token-based models, leaving T3GG and T1GG in the second and the third place, respectively. On average, the token-based models consistently outperform the character-based ones with respect to both effectiveness and efficiency. Again, this should be attributed to the comprehensive, noise-free content of curated documents. Nevertheless, the accuracy of C4GG over SVM is practically equivalent to T1GG, the most accurate token-based model for this learner.

Comparing bag with graph models, we infer from Table 7 that the former are more efficient by two orders of magnitude across all models, despite the disproportionate difference in their dimensionality (cf. Table 6). The reason is the higher computational cost of feature extraction: the bag models are efficiently implemented with the help of inverted indices, while the graph ones involve comparisons with large graphs. Yet, the few extracted graph similarities convey more information than the numerous features of bag models. To this attests their relative performance over NBM, where the former are more accurate than the latter by at least 15 %. The advanced learning of SVM, though, leverages the character-based

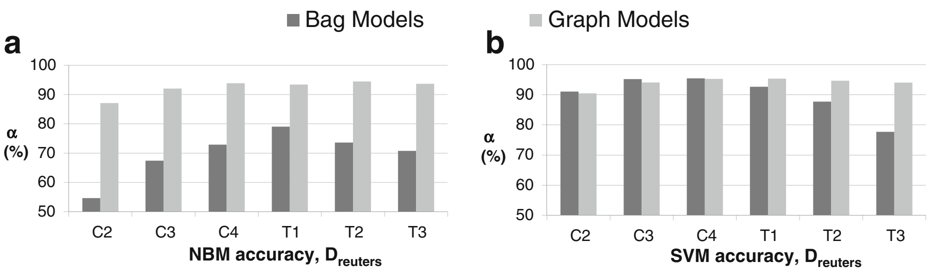

Figure 5 The relative classification accuracy of bag and graph models with respect to (**a**) NBM and (**b**) SVM over $D_{reuters}$

Table 7 The classification time t_c in milliseconds (ms) for each dataset and model

		C2	C3	C4	T1	T2	T3
$D_{reuters}$	Bag models	0.5	1.0	1.4	0.3	0.4	0.6
	Graph models	92.3	407.6	649.0	145.0	54.8	7.4
D_{blogs}	Bag models	0.2	0.4	0.5	0.1	0.1	0.1
	Graph models	106.3	660.6	715.8	119.6	60.9	47.4
$D_{twitter}$	Bag models	0.2	0.3	0.4	0.1	0.1	0.1
	Graph models	46.4	98.4	96.6	30.2	9.2	7.9

bag models to a significantly higher accuracy than their graph counterparts (the token-based graph models maintain their superiority over the bag ones even with SVM). This phenomenon should be attributed to the lack of sparseness and noise in curated documents (challenges Ch2 and Ch3).

On the whole, we can conclude that applications emphasizing effectiveness should combine SVM with either type of model, since the maximum accuracy of bag and graph models is identical in practice (it amounts to 95.46 % and 95.35 % for C4G and T1GG, respectively). When a weak learner is employed, T2GG significantly outperforms all other models. The best balance between effectiveness and efficiency is offered by T1G in conjunction with SVM.

Semi-curated documents Figure 6a and b depict the accuracy of all models over D_{blogs} when coupled with NBM and SVM, respectively. We notice that the bag models exhibit similar patterns as in the case of $D_{reuters}$: higher core sizes increase the accuracy and the classification time of the character-based models, while having the opposite effect on the accuracy of the token-based ones (their t_c remains intact). In general, the average accuracy of character-based models is higher than the token-based ones for both classification algorithms, with the difference increasing over SVM. This is a direct consequence of the noisy, non-standard vocabulary in semi-curated documents (challenges Ch2 and Ch4). However, the higher effectiveness of character-based models comes at the cost of lower time efficiency: a double t_c in the best case.

For graph models, we observe again a disparity between the two model granularities. Larger core sizes increase the accuracy and the classification time of character-based models, while decreasing t_c for the token-based ones. The accuracy of the latter exhibits an inconsistent behavior across the two classifiers: T2GG and T1GG are the most and the

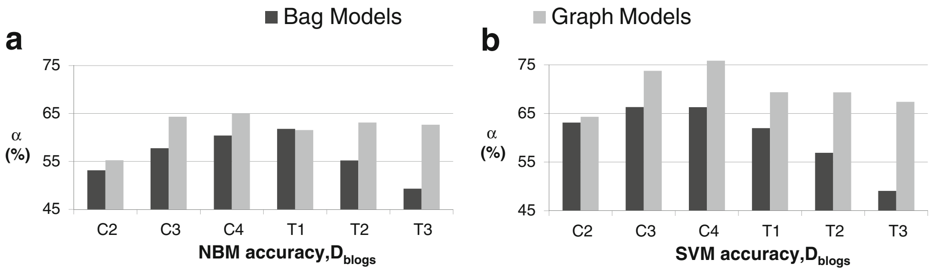

Figure 6 The relative classification accuracy of bag and graph models with respect to (**a**) NBM and (**b**) SVM over D_{blogs}

least accurate models over NBM, respectively, but they coincide to the maximum accuracy, when coupled with SVM. On average, the token-based models are considerably more efficient than the character-based ones, while being more accurate, as well, when combined with NBM. This should be attributed to the lack of sparseness in semi-curated documents (challenge Ch3). However, the character-based models hold a statistically significant lead in accuracy over SVM. With $\alpha = 75.9$ %, C4GG actually achieves the overall maximum accuracy, surpassing its bag counterpart by almost 10 % – at the cost of a substantially lower efficiency.

Similarly, most graph models outperform their bag counterparts across both classification algorithms, with the differences being more intense for token-based models. Only in two cases is this difference insignificant: T1 over NBM and Ch2 over SVM. The superiority of graph models should be attributed to the noisy and non-standard vocabulary of the semi-curated documents (challenges Ch2 and Ch4). Therefore, the combination of SVM with graph models, especially C4GG, should be preferred by applications emphasizing effectiveness. On the flip side, the classification time of graph models is higher by two to three orders of magnitude.

Raw documents Figure 7a and b present the accuracy of all models over $D_{twitter}$ in combination with NBM and SVM, respectively. We observe that the accuracy of bag models exhibits a different behavior from the other datasets: larger core sizes significantly decrease α in all cases, except for character-based models in conjunction with NBM. The difference between C2G and C4G over NBM is actually the smallest one across all datasets (<7 %), while the situation is reversed for SVM, with the former significantly outperforming the latter by 3 %. We also observe that the average accuracy of character-based models is much larger than the token-based ones across both learners. These patterns highlight the advantages of character n-grams in the context of sparseness and of noisy terms, which stem from the abbreviations and neologisms that abound in raw documents (challenges Ch2, Ch3 and Ch4).

New patterns are also observed in the accuracy of graph models: $D_{twitter}$ is the only dataset for which higher core sizes significantly decrease the accuracy of token-based models. Apparently, this condition should be attributed to the noisy, short neologisms and the sparseness of raw documents (challenges Ch2, Ch3 and Ch4). In contrast, the character-based models are inherently crafted for these settings and increase their accuracy with higher core sizes. They actually outperform the token-based ones across both classifiers, on average. Their most accurate model is C4GG, regardless of the classification algorithm (its difference from C3GG over SVM is just 0.4 %, but it is statistically significant). This is a

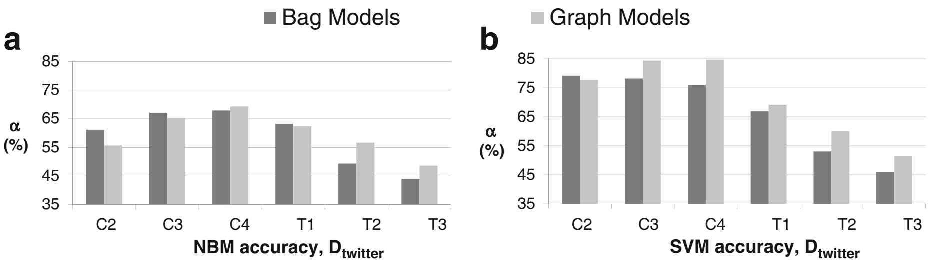

Figure 7 The relative classification accuracy of bag and graph models with respect to (**a**) NBM and (**b**) SVM over $D_{twitter}$

consistent pattern across all datasets, but this is the only document type, where the relative performance of character-based graph models differs from their bag counterparts: C2G and C2GG are the most and the least accurate character-based models in conjunction with SVM, respectively.

This pattern actually suggests that the graph similarities of bigram graphs are less discriminative than the TF weights of character bigrams. In other words, the co-occurrence frequency of pairs of bigrams is less discriminative than the frequency of their individual occurrence. This should be attributed to the low number of bigrams, which restricts their pairwise combinations. In contrast, the character trigrams and four-grams have a higher dimensionality by 2 and 7 times, respectively (see Table 6). This leads to a quadratically higher number of pairwise combinations and, thus, more distinctive edges, which increase the discriminativeness of C3GG and C4GG. Note that this restriction does not apply to token-based graph models, since there is no limit to the number of tokens n-grams they consider. The limited dimensionality of the corresponding bag models in Table 6 emanates from the scarceness of token n-grams and not from the limited number of their combinations.

Comparing bag with graph models, we notice that the former are significantly more accurate than the latter in just four cases: C2 across both algorithms, a situation that was explained above, and C3 and T1 over NBM. For the last two cases, the correlations are reversed over SVM, thus indicating that the graph similarities produced by C3GG and T1GG suffer from so low discriminativeness that NBM cannot exploit them. In all other cases, the graph models significantly outperform their bag counterparts. With respect to efficiency, we observe that the classification time of character-based models increases with higher core sizes; for the token-based models, t_c is either insensitive to n (bag models) or decreases with its increase (graph models). Again, the higher accuracy of graph models comes at the cost of significantly lower efficiency.

On the whole, we can conclude that the token-based models are inadequate for handling raw documents, due to the high levels of noise and sparseness. Instead, applications emphasizing effectiveness should combine SVM with character-based graph models and C4GG, in particular. The best balance between effectiveness and efficiency is offered by C2G in conjunction with SVM.

5.6 Discussion

We now summarize the main findings of our experimental study. Starting with the graph models, we verified that the merge portion has a decisive role in their balance between effectiveness and efficiency – regardless of the document type. In most cases, their classification time is significantly enhanced by constructing the class graphs with less than half the labelled documents of the corresponding topic ($mp < 0.5$). Given that t_c is dominated by comparisons with class graphs, the smaller they are, the lower t_c gets. At the same time, the impact on classification accuracy is limited, thus implying that the class graphs are able to extract distinguishing topic patterns from a portion of the training set.

Another important factor for the efficiency of graph models is their granularity. In fact, the token-based models are substantially more efficient than the character-based ones, due to the smaller size and order of their class graphs. Higher core sizes yield more nodes (token n-grams), but less edges, thus improving their classification time. For this reason, T3GG is the most efficient graph model by far across all document types. For the character-based models, higher core sizes increase both the size and the order of class graphs. As a result, C4GG consistently ranks as the most time-consuming graph model.

The effectiveness of graph models is mainly determined by the type of their features as well as their core size. With respect to the first factor, we verified that the hybrid features combine the strengths of numerical and nominal ones, exhibiting high robustness at no cost in efficiency. The second factor, the core size, is more decisive in the case of character-based graph models. C2GG consistently achieves the lowest accuracy among them, as it involves a limited number of nodes (character bigrams) that are densely connected across all topics. Combined with NBM, C2GG is actually the least effective across all graph models – the only exception being T3GG for $D_{twitter}$. In contrast, C4GG achieves the maximum accuracy across all document types, when combined with SVM. Among the token-based models, T1GG and T2GG take turns at the maximum accuracy, while T3GG is the least effective, due to the sparseness of its class graphs.

The document type constitutes an external factor that affects both the efficiency and the effectiveness of graph models. The most important factors for efficiency are the average document size and the number of topics it involves. We can assess their impact on classification time judging from the behavior of C2GG, the only model that uses the same merge portion across all datasets. Comparing D_{blogs} with $D_{twitter}$, we observe that its t_c drops by 57 % as the document size decreases by 88 %, from 1,100 characters to just 130. Comparing $D_{reuters}$ with D_{blogs}, we notice that its t_c increases by 13 % even though the document size decreases by 8 %. The reason is that the number of classes rises from 4 to 10 and the number of features from 31 to 166. This behavior suggests that the effect of document size on the efficiency of class graphs is larger than the effect of the number of topics and the dimensionality of the feature space. Note, though, that the document type does not affect the relative efficiency of graph models: higher core sizes increase t_c for the character-based models across all datasets (the only exception is C3GG and C4GG over $D_{twitter}$) and vice versa for the token-based ones.

In contrast, the document type plays a decisive role for the relative effectiveness of character- and token-based graph models. This is illustrated by their average accuracy in conjunction with NBM, which is more sensitive to the quality of the input features than SVM. The curated documents of $D_{reuters}$ promote the accuracy of token-based models, which outperform the character-based ones by 3 %, on average. The smaller document sizes and the higher levels of noise in D_{blogs} decrease the average distance of the two granularities to less than 1 %. This situation is totally reversed for the raw documents of $D_{twitter}$, with the character-based models taking an average lead of 7.5 %. Similar but more intense patterns are exhibited by the bag models in combination with NBM: the lead of token-based ones starts from 13 % for $D_{reuters}$, it is reduced to 1.5 % for D_{blogs} and turns negative (-13 %) for $D_{twitter}$. We can conclude, therefore, that the token-based models provide a better balance between effectiveness and efficiency than the character-based ones only for curated documents.

The document type is also decisive for the relative effectiveness of bag and graph models. In conjunction with NBM, the latter are more accurate than their bag counterparts by 23 %, 6 % and 1 %, on average, over $D_{reuters}$, D_{blogs} and $D_{twitter}$, respectively. This indicates that the superiority of edge weights over the weights of individual n-grams decreases proportionally with the level of noise. In other words, the more noisy the content of a document type, the closer the similarities of an unlabelled document with graphs of different classes and the less distinctive the graph similarities from the individual n-grams. However, the advanced learning of SVM exploits the graph similarities to a larger extent, giving them the lead over bag models in most cases. The best graph model is actually more accurate than the best bag one over D_{blogs} and $D_{twitter}$ by 9.5 % and 5.5 %, respectively (in the case of $D_{reuters}$, there is no practical difference).

Regarding efficiency, the document type does not affect the relative performance of bag and graph models. The former achieve a consistently lower classification time by at least two orders of magnitude across all datasets – despite the significantly lower dimensionality of graph models, as indicated in Table 6. Note, though, that the classification time of graph models can be significantly reduced through a simple parallelization scheme that assigns each class graph to an independent node.

On the whole, the graph models are particularly effective for semi-curated and raw documents, while their lower dimensionality conveys two additional, qualitative benefits over bag models. First, they yield simple classification models that are interpretable, provided that the corpus involves a limited number of topics. In contrast, many classification algorithms like decision trees are hard to train with bag models. Second, the graph models implicitly apply dimensionality reduction and, thus, no further reduction or transformation of the feature space is required. In this way, the n-gram graphs facilitate the configuration of relevant applications and reduce their complexity.

Finally, it is worth noting that, so far, we have implicitly assumed that every corpus is *homogeneous*, containing documents that exclusively belong to a single document type. In practice, though, it is possible to come across a *heterogeneous corpus*, which involves a mixture of curated, semi-curated and raw documents. In such cases, the optimal representation model can be selected based on the extent of type heterogeneity and the application requirements. If the corpus is dominated by a specific type (i.e., it accounts for more than 80 %–90 % of all documents), our experimental results can be used as guidelines for selecting the best model for this type with respect to time efficiency or classification accuracy. In both cases, a small penalty in accuracy should be expected, due to the documents of different types. For corpora that cover two or three document types to a significant extent, the overall best model should be applied. This is the character four-grams graph model (C4GG) in conjunction with the hybrid features and SVM for applications emphasizing classification accuracy; for all document types, it consistently achieves either the highest or one of the top accuracies against all other models, both graph and bag ones. In case time efficiency is more important than effectiveness, the character four-grams bag model (C4G) in combination with SVM seems to be a reliable solution.

6 Related work

Text classification has been extensively studied for several years either as a stand-alone domain [36] or as part of text mining research [3]. Its domain has evolved to cover a variety of different classification settings, ranging from topic classification [22] and spam detection (e.g., [24]) to genre and author detection [42].

In every setting, the document representation model plays a crucial role and, thus, it should be carefully selected. As we have already stressed in Section 3, the usual choice are the bag models [22, 36]. However, they are not the most common representation, when moving from topic classification to other sub-domains of text classification. In fields like author identification [40] and genre classification, a whole new set of features has been devised and used. Such features may rely on syntactic, grammatical and morphological information [40, 42], but also on sub-word character sequences [39] (possibly in conjunction with string kernel functions [28]) as well as higher-order token-based models [23, 32].

Another graph-based representation model that has been proposed in the literature is the Universal Networking Language (UNL) [5, 6]. It transforms a document into a graph, where the nodes correspond to words and the edges denote a semantic relation between the adjacent words. A UNL graph is then transformed into a feature vector by converting every node into a dimension that is weighted according to the corresponding node degree. This approach, however, is fundamentally different from the n-gram graphs. Unlike UNL, their representation acts on the morphology of the language. It is dependent on symbols (characters) and no preprocessing steps are required to form the n-gram graphs. This is because the edges indicate a single relation that is not semantic and, thus, it can be asserted without any language-specific knowledge, simply based on the neighborhood of two n-grams.

Some works have focused on the relative performance of the established representation models. None of them, though, considers n-gram graphs, while their scope is limited, compared to our analysis, focusing on a specific application of text classification. In [41], the author compares character n-grams and token unigrams in the context of *author attribution* (i.e., the task of identifying the author of a document). The experimental outcomes suggest that character n-grams yield larger feature spaces, but provide more robust features, when the training and the testing sets are heterogeneous in terms of author distributions and thematic areas.

In [9], the authors examine four content-based representation models: lemmatized token unigrams, lemmatized token bigrams and lemmatized "dependency triples"[15] that were obtained from the Stanford and the AEGIR parsers. The comparative analysis was performed over a set of curated documents that contained patent abstracts in English. The outcomes suggest that the combination of all four representations achieves the best improvement over the baseline (the lemmatized token unigrams) and that the lemmatized token bigrams account for the most important contribution to this improvement. However, the authors stress that their outcomes are language-dependent, since similar experiments over French and German abstracts yielded different results. They actually conclude that there is no benefit in using token bigrams for compounding languages like German, which express complex concepts with a single word.

Another issue of text classification that has been widely examined in the literature is sparseness. Most techniques for dealing with it incorporate contextual knowledge into the content-based representation models. For instance, the authors in [43] enrich the representation of text snippets (from advertisements or tweets) using "explicit semantic analysis" to map them to Wikipedia concepts. The metadata of linked objects appearing in forum posts are an alternative source of external data; they can improve the topic classification of short texts even if they are used without the original features [25]. Another approach advocates the use of search engines: the content of a raw document d_i is submitted as a query to a search engine and the resulting Web pages are crawled to build a bag representation model for d_i from their terms [30]. In other lines of research, the classification of short messages was improved by augmenting their features with author-related information [38] and by performing topic modelling after aggregating them [20]. However, all these approaches are typically application-dependent and, thus, lie out of the scope of this work.

[15] A "dependency triple" is a language-dependent feature comprising two words that are semantically connected with one of the syntactic relators that are supported by the corresponding parser. For example, $subj(Y, X)$ denotes a feature consisting of a noun Y that is connected with a verb X through the relator "subject".

An alternative solution to sparseness is the representation of documents in a vector space of “latent topics”. These can be derived from dimensionality reduction methods, such as Latent Semantic Indexing (LSI) [8] and Latent Dirichlet Allocation (LDA) [4]. For example, the authors in [33] represent short texts by a combination of terms and hidden topics that were extracted from an external “universal dataset” using LDA. Given a good universal dataset and the right number of hidden topics, the expanded vector space was found to significantly improve classification performance. In [47], the authors improve the classification of short texts using the Transductive LSI, which extracts information from the test instances. We do not elaborate on such dimensionality reduction methods within this work, because we focus on the base representation itself. Besides, LSI and LDA differ substantially from the n-gram graphs in that they are applied on a token-based basis, while our approach is applied to the graph world (taking into account token or character proximity). Additionally, LDA differs from our approach in that it uses exogenous information (i.e., the number of hidden topics) and incorporates probabilistic methods in the representation. In contrast, the extraction of the class n-grams is deterministic and straightforward, with no need to converge.

On another line of research, many works on text classification examine the particular challenges of individual domains, especially those posed by the user-generated content of Social media, like Twitter. UGC actually offers new research grounds, where many of the intricacies of the classification task are present: multi-lingual, voluminous content, very short texts, fully evolving, non-standard vocabulary, noise and lack of labelled resources. In [46], the authors describe a large-scale system that aligns a stream of tweets to a predefined ontology in real-time and with high precision. It uses a topic modelling approach that combines token unigrams and character four-grams with exogenous features, such as the content of Web pages that correspond to embedded URLs. [34] addresses Topic Detection[16] in the context of Twitter using fuzzy fingerprints: every topic (i.e., hashtag) is represented by the set of its top-k most frequent token unigrams, which are used to estimate its similarity with every new, unclassified tweet. A system for “trend stuffing” is proposed in [21]; it employs a binary classification scheme to decide whether a tweet is related to a highly active topic (“trend”) so as to facilitate the detection of spam tweets. In [27], the authors use text classification for keyword extraction from “social snippets”, such as status updates, interesting events or recent news. In [2], tweets are classified into positive, negative and neutral ones according to their sentiment. Finally, in [16], the authors measure the semantic relatedness between pairs of tweets using external knowledge: they map each tweet to Wikipedia and then exploit the links between Wikipedia pages. All these tasks offer promising, future applications for the n-gram graphs approach we examined in this work.

7 Conclusions

In this work, we demonstrated that the n-gram graphs offer a noise-tolerant, language-neutral representation model for Topic Classification that is inherently capable of addressing sparseness. We elaborated on the parameters of this approach and fine-tuned them through a thorough experimental study. The most important one is the “merge portion”, which specifies what part of the labelled documents participates in the construction of class graphs.

[16]*Topic Detection* is similar to Topic Classification, but differs in that it involves many more classes, which are also so rare that an unlabelled document is likely to belong to none of them [34].

In most cases, small values (< 50 %) sacrifice effectiveness to a negligible extent for significantly higher efficiency. Another crucial parameter is the type of features, which can be numeric, nominal or hybrid. Among them, hybrid features were found to offer the most robust performance. The other two parameters are the "model granularity" (i.e., character- or token-based) and the "core size" (n), which apply to bag models, as well. The optimal configuration of both parameters depends on the type of Web documents. To identify the type of a corpus, we introduced four metrics for quantifying its main characteristics. We also proved that this external parameter affects the relative performance of graph and bag models. In general, though, the graph models were found to achieve significantly higher classification accuracy than their bag counterparts, at the cost of lower efficiency.

In the future, we intend to examine parallelization techniques for enhancing the time efficiency of n-gram graphs. We actually plan to adapt the functionality of two procedures to the MapReduce framework: the creation of class graphs and the comparisons between class and document graphs. Given that the class n-gram graphs may be viewed as an intermediate dimensionality reduction step, we also plan to examine their relation to LSI and LDA.

References

1. Amini, M.R., Usunier, N., Goutte, C.: Learning from multiple partially observed views - an application to multilingual text categorization. In: NIPS, pp. 28–36 (2009)
2. Batista, F., Ribeiro, R.: Sentiment analysis and topic classification based on binary maximum entropy classifiers. Proc. Leng. Nat. **50**, 77–84 (2013)
3. Berry, M.W., Kogan, J.: Text Mining: Applications and Theory. Wiley, Chichester (2010)
4. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
5. Choudhary, B., Bhattacharyya, P.: Text clustering using semantics. World Wide Web Conference (2002)
6. Choudhary, B., Bhattacharyya, P.: Text clustering using universal networking language representation. World Wide Web Conference (2002)
7. Dasgupta, A., Drineas, P., Harb, B., Josifovski, V., Mahoney, M.W.: Feature selection methods for text classification. In: KDD, pp. 230–239 (2007)
8. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
9. D'hondt, E., Verberne, S., Koster, C.H.A., Boves, L.: Text representations for patent classification. Comput. Linguist. **39**(3), 755–775 (2013)
10. Dumais, S., Chen, H.: Hierarchical classification of web content. In: SIGIR, pp. 256–263 (2000)
11. Eisenstein, J., O'Connor, B., Smith, N.A., Xing, E.P.: A latent variable model for geographic lexical variation. In: EMNLP, pp. 1277–1287 (2010)
12. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
13. Figueiredo, F., Belém, F., Pinto, H., Almeida, J.M., Gonçalves, M.A., Fernandes, D., de Moura, E.S., Cristo, M.: Evidence of quality of textual features on the web 2.0. In: CIKM, pp. 909–918 (2009)
14. Forman, G.: An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. **3**, 1289–1305 (2003)
15. Garcia Esparza, S., O'Mahony, M., Smyth, B.: Towards tagging and categorization for micro-blogs. In: AICS (2010)
16. Genc, Y., Sakamoto, Y., Nickerson, J.V.: Discovering Context: Classifying Tweets through a Semantic Transform Based on Wikipedia, pp. 484–492 (2011)
17. Giannakopoulos, G., Karkaletsis, V., Vouros, G.A., Stamatopoulos, P.: Summarization system evaluation revisited: N-gram graphs. TSLP **5**(3), 1–39 (2008)
18. Giannakopoulos, G., Palpanas, T.: Content and type as orthogonal modeling features: a study on user interest awareness in entity subscription services. Int. J. Adv. Netw. Serv. **3**(2), 296–309 (2010)
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: An update. ACM SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)

20. Hong, L., Davison, B.: Empirical study of topic modeling in twitter. In: SOMA, pp. 80–88 (2010)
21. Irani, D., Webb, S., Pu, C., Li, K.: Study of trend-stuffing on twitter through text classification. In: CEAS, pp. 40–49 (2010)
22. Joachims, T.: Text categorization with suport vector machines: Learning with many relevant features. In: ECML, pp. 137–142 (1998)
23. Keselj, V., Peng, F., Cercone, N., Thomas, C.: N-gram-based author profiles for authorship attribution. In: PACLING, pp. 255–264 (2003)
24. Khorsi, A.: An overview of content-based spam filtering techniques. Informatica **31**, 269–277 (2007)
25. Kinsella, S., Passant, A., Breslin, J.G.: Topic classification in social media using metadata from hyperlinked objects. In: ECIR, pp. 201–206 (2011)
26. Kinsella, S., Wang, M., Breslin, J.G., Hayes, C.: Improving categorisation in social media using hyperlinks to structured data sources. In: ESWC (2), pp. 390–404 (2011)
27. Li, Z., Zhou, D., Juan, Y.F., Han, J.: Keyword extraction for social snippets. In: WWW, pp. 1143–1144 (2010)
28. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. **2**, 419–444 (2002)
29. Manning, C., Raghavan, P., Schuetze, H.: Introduction to information retrieval, vol. 1. Cambridge University Press (2008)
30. Meng, W., Lanfen, L., Jing, W., Penghua, Y., Jiaolong, L., Fei, X.: Improving short text classification using public search engines. In: Integrated Uncertainty in Knowledge Modelling and Decision Making, pp. 157–166 (2013)
31. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining, pp. 1320–1326. LREC (2010)
32. Peng, F., Schuurmans, D.: Combining naive bayes and n-gram language models for text classification. Advances in Information Retrieval, pp. 547–547 (2003)
33. Phan, X.H., Nguyen, M.L., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: WWW, pp. 91–100 (2008)
34. Rosa, H., Batista, F., Carvalho, J.P.: Twitter topic fuzzy fingerprints. In: IEEE International Conference on Fuzzy Systems, pp. 776–783 (2014)
35. Salton, G.: The Smart Retrieval System – Experiments in Automatic Document Processing, p. 556. Prentice-Hall (1971)
36. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002)
37. Sebastiani, F.: Text categorization. In: Encyclopedia of Database Technologies and Applications, pp. 683–687 (2005)
38. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M.: Short text classification in twitter to improve information filtering. In: SIGIR, pp. 841–842 (2010)
39. Stamatatos, E.: Ensemble-based author identification using character n-grams. In: Proceedings of the 3rd International Workshop on Text-based Information Retrieval, pp. 41–46 (2006)
40. Stamatatos, E.: A survey of modern authorship attribution methods. J. Am. Soc. Inf. Sci. Technol. **60**(3), 538–556 (2009)
41. Stamatatos, E.: On the robustness of authorship attribution based on character n-gram features. J. Law Policy **21**(2), 421–439 (2013)
42. Stamatatos, E., Fakotakis, N., Kokkinakis, G.: Automatic text categorization in terms of genre and author. Comput. Linguist. **26**(4), 471–495 (2000)
43. Sun, X., Wang, H., Yu, Y.: Towards effective short text deep classification. In: SIGIR, pp. 1143–1144 (2011)
44. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, p. 560. Morgan Kaufmann, San Francisco (2005)
45. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: WSDM, pp. 177–186 (2011)
46. Yang, S., Kolcz, A., Schlaikjer, A., Gupta, P.: Large-scale high-precision topic modeling on twitter. In: KDD, pp. 1907–1916 (2014)
47. Zelikovitz, S., Hirsh, H.: Transductive lsi for short text classification problems. In: FLAIRS, pp. 556–561 (2004)