UTILIZANDO SIMILARIDADE SEMÂNTICA PARA APRIMORAR A REPRESENTAÇÃO DE DOCUMENTOS TEXTUAIS

VICTOR SILVA RODRIGUES

UTILIZANDO SIMILARIDADE SEMÂNTICA PARA APRIMORAR A REPRESENTAÇÃO DE DOCUMENTOS TEXTUAIS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Marcos André Gonçalves

Belo Horizonte

Junho de 2018

VICTOR SILVA RODRIGUES

EXPLOITING SEMANTIC SIMILARITY FOR IMPROVED TEXT REPRESENTATION

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Marcos André Gonçalves

Belo Horizonte

June 2018

Rodrigues ,Victor Silva

R696e Exploiting semantic similarity for improved text representation [manuscrito] / Victor Silva Rodrigues. - 2018.
xv, 50 f. il.

Orientador: Marcos André Gonçalves
Dissertação (mestrado) - Universidade Federal de Minas Gerais;Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.46-49
.
1. Computação – Teses. 2. Indexação automática – Teses. 3. Processamento da linguagem natural (Computação) – Teses I. Gonçalves, Marcos André. II. - Universidade Federal de Minas Gerais;Instituto de Ciências Exatas, Departamento de Ciência da Computação. III.Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Exploiting Semantic Similarity for Improved Text Representation

VICTOR SILVA RODRIGUES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Marcos André Gonçalves - Orientador
Departamento de Ciência da Computação - UFMG

Profa. Gisele Lobo Pappa
Departamento de Ciência da Computação - UFMG

Prof. Mario Sérgio Ferreira Alvim Júnior
Departamento de Ciência da Computação - UFMG

Prof. Leonardo Chaves Dutra da Rocha
Departamento de Ciência da Computação - UFSJ

Belo Horizonte, 24 de agosto de 2018.

Acknowledgments

A conclusão do mestrado em Ciência da Computação é um grande marco em minha vida. Como todo grande marco, ele não seria possível sem o amparo de algumas pessoas e instituições. Gostaria de deixar minha nota de gratidão a todos que tornaram este momento possível.

Aos meus pais, Marilda e Gasparino, por seu amor incondicional, e por sempre terem me incentivado a estudar, fornecendo todas as condições para que eu pudesse priorizar minha educação. Aos meus irmãos, Érica e Emerson, que sempre estiveram ao meu lado. É um privilégio muito grande tê-los em minha vida.

Ao meu orientador Marcos Gonçalves, que, além de promover discussões produtivas em tornos de ideias a serem perseguidas e aprimoradas, foi meu mentor acadêmico desde o início, em minha primeira iniciação científica. Aos meus colegas de laboratório, em especial Thiago Salles e Sérgio Canuto, que tiveram um papel ativo na construção do conhecimento na área de pesquisa em que atuo.

A todos os meus amigos, que me acompanharam por essa jornada e me renderam horas de conversas e momentos inigualáveis. Em especial a Luiz, Gabriel e Nildo. Ao Murilo, cuja companhia sempre me impulsionou a seguir em frente.

Aos meus tios e tias, cujas histórias são uma grande inspiração, por estarem sempre presentes, formando uma rede de apoio durante minha jornada. Gostaria de agradecer especialmente a minha tia paterna Elizabeth (in memoriam), que nos doou o primeiro computador a que tive acesso em casa, e que proporcionou a primeira faísca do meu interesse por informática e computação.

Finalmente, agradeço a todos os professores que fizeram parte da minha formação, ao DCC, à UFMG, e ao programa Ciência sem Fronteiras por terem permitido expandir meus horizontes pessoais, profissionais e acadêmicos.

"A wise man proportions his belief to the evidence."
(David Hume)

Resumo

A Classificação Automática de Documentos é uma técnica fundamental quando se trata da extração de informações úteis da grande e crescente quantidade de dados textuais produzidos diariamente na Internet e dentro das organizações. Recentemente, Vetores de Palavras (*Word Embeddings*, como por exemplo Word2Vec) foram propostos para representar termos como vetores cujas similaridades correspondem à proximidade semântica entre as palavras. Além disso, existem linhas de pesquisa cujo objetivo é compreender a utilização de Vetores de Palavras para melhorar a classificação textual. Entretanto, os resultados atuais dependem de muitos ajustes finos em suas parametrizações, e seus resultados nem sempre são consistentes quanto à superioridade em relação ao modelo tradicional de Saco-de-Palavras (*Bag-of-Words*). Como as palavras mais próximas em um modelo de Vetores de Palavras são semanticamente relacionadas, propomos um novo método de geração de atributos a partir de agrupamentos de palavras similares. Nós nos referimos a esses agrupamentos como "hyper-palavras" (*hyperwords*), uma vez que eles correspondem a novos conceitos semânticos, mais ricos do que as palavras simples. Nós propomos, ainda, uma adaptação ao modelo TF-IDF de assinalamento de pesos, criado especificamente para as hyper-palavras, que pode ser utilizado de forma similar àquela utilizada pelos termos originais, efetivamente substituindo as palavras na representação de documentos. Demonstramos que os atributos gerados a partir de hyper-palavras são significativamente mais discriminativos do que aqueles obtidos a partir de palavras simples. Também experimentamos uma combinação entre os atributos de hyper-palavras com os atributos derivados de uma técnica estado-da-arte de agregação de vetores de palavras, obtendo um método robusto. Experimentos amplos foram executados utilizando 24 bases de comparação em classificação de tópicos e de análise de sentimentos, comparando com métodos estado-da-arte em vetores de palavras, demonstrando a superioridade da nossa proposta em grandes margens, obtendo ganhos de até 18% em classificação de tópicos e 16% em classificação de sentimentos quando comparado ao modelo de Saco-de-Palavras.

Palavras-chave: Classificação Automática de Documentos, Hyper-palavras, Saco-de-Palavras, Vetores de Palavras.

Abstract

Automatic Document Classification is a key technique to help extracting useful information from the huge amount of textual data produced daily on the Web and inside organizations. Recently, Word Embeddings (e.g., Word2Vec) have been proposed for representing terms as vectors whose similarities should correlate with semantic relatedness. There has also been some research on how to use Word Embeddings to improve text classification. Nevertheless, current results depend on heavy and careful parameter tuning and still do not consistently outperform Bag-of-Words representation in a variety of scenarios. Since the nearest words of a given Word Embedding w are all semantically related to each other, we propose a new method for generating features from clusters of similar Word Embeddings. We refer to these clusters as *hyperwords*, since they correspond to new semantic concepts, richer than simple words. We propose an adaptation of the TF-IDF weighting scheme for these new features so that they can be used similarly to the original terms, but substituting them. We demonstrate that features generated from hyperwords are significantly more discriminative than those obtained from simple words. We also experiment with the combination of the hyperwords-based representation with a state-of-art pooling technique, obtaining a very robust method. Extensive experiments performed using 24 benchmarks on topic classification and sentiment analysis against state-of-the-art baselines that exploit Word Embedding-based document representations show the superiority of our proposals by large margins, achieving gains up to 18% on topic classification datasets and 16% in sentiment classification datasets over the Bag-of-Words representation.

Keywords: Text Classification, Hyperwords, Bag-of-Words, Word Embeddings.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this chapter, we introduce the motivation of this work, our proposals for improvement, the research questions to be answered and the general organization of the remaining chapters in this dissertation.

1.1 Motivation

The amount of data that is created and shared everyday has never been as large as it is today. In light of so much data being constantly created, methods for categorizing documents automatically are of extreme importance, now more than ever. A common area of study that deals directly with this problem is named in the literature as Automatic Document Classification.

Automatic Document Classification methods usually rely on the supervised learning paradigm [Sebastiani, 2002], where a classification model is learned using manually labeled documents (training set), and then used to classify the unseen documents (testing set). There are many ways in which documents can be represented. In other words, there are many ways of mapping documents to a feature set to be used in a supervised classification model. The most popular way of representing a collection of documents exploits a fixed-length vector representation, referred to in the literature as the Bag-of-Words representation (BoW) [Harris, 1954]. In this representation, each vector is of length equal to the size of the collection's vocabulary, and each element of the vector has a value corresponding to the weight of the term in the document which the vector represents. There exist several robust strategies to weight the importance of terms (e.g. TF, IDF, TF-IDF [Salton and Buckley, 1988]).

Despite its simplicity and low computational cost, the Bag-of-Words representation suffers with problems in sparse collections, as most documents only contain a small

fraction of the whole vocabulary, making it easy for documents that are semantically related have few or no features in common, specially for short documents. Moreover, the BoW representation usually does not keep any information about the positions occupied by the words in the original document, which can often be correlated with semantic relationships among terms.

Recent efforts for extracting more meaningful semantic information from text have culminated in the creation of an efficient and effective word embedding model, called Word2Vec [Mikolov et al., 2013a]. Word2Vec is a shallow neural network model whose objective is predict a word given its context. In essence, it captures co-occurrence relationships between words in a given context window. Thus, words that occur in similar contexts have similar vector representations, i.e., embeddings. This allows one to deal with richer semantic relationships in which words are represented as low-dimensional and latent vectors (e.g. 200 dimensions) whose relative similarities correlate with semantic relatedness.

1.2 Our Proposal and Research Questions

In this dissertation, we propose a new method for document representation that takes advantage of the simplicity and effectiveness of the Bag-of-Words model without ignoring the potential benefits of considering the semantics of words provided by Word Embeddings. More specifically, we hypothesize that it is possible to group words related to a similar semantic concept into a single feature in the vector space model, which we call *hyperword* (i.e., a cluster of "similar" words) and generalize the known TF-IDF statistics to assign proper weights to hyperwords.

To generate these hyperwords, we expand each word in the dataset's dictionary with its closest neighbors in the Word2Vec representation. By controlling the minimum similarity of the closest neighbors, we are able to set the specificity level of the semantic concept corresponding to each hyperword. For example, when expanding the word "good", we may generate the hyperword "good, great", which refers specifically to good concepts. On the other hand, the hyperword represented by the cluster "good, great, bad, terrific" is related to a more general concept of qualifying entities. This controlling mechanism provides flexibility to the specificities of each classification task (e.g. topic categorization versus sentiment analysis), being useful to ease the shortage of information provided by particularly rare words. Moreover, we introduce a variant of the TF-IDF weighting scheme that explicitly takes into consideration the similarity measure used to build the hyperwords, emphasizing the semantic relationships among

words inside a hyperword and re-defining the relative importance of terms belonging to different hyperwords.

In summary, the main research questions we address in this dissertation are:

RQ1: Can the Bag-of-Words model be reformulated to benefit from the information encoded in a Word Embeddings model, such as the Word2Vec?

RQ2: Is our proposal of clustering Word Embeddings to build semantic concepts effective when compared to the traditional Bag-of-Words and other state-of-the-art strategies based on Word Embeddings?

RQ3: Are hyperwords more discriminative than single words? In other words, can hyperwords be used to improve the quality of text representation, when compared to single words?

In order to answer these questions, we have performed a series of experiments using a variety of sentiment classification (19 datasets) and topic classification (5 datasets) benchmarks. We evaluate different document representations using two strong machine learning methods: the SVM and the Random Forest classifiers. Our experimental results demonstrate that the effectiveness of the proposed document representation is not only much superior than the traditional BoW representation [Harris, 1954] but also superior to other recent document representations based on Word Embeddings [Lev et al., 2015; Le and Mikolov, 2014; Tang et al., 2015]. In particular, our approach achieved gains up to 8% in macro averaged F_1 and up to 18% in micro averaged F_1 on topic datasets over the BoW representation. Also, our document representation performed consistently better than the state-of-the-art word embeddings (Fisher pooling method [Lev et al., 2015], Predictive Text Embeddings [Tang et al., 2015]), with gains up to 11% in micro averaged F_1. It also performed consistently better in all datasets about sentiment classification, achieving gains up to 16% in macro averaged F_1. In addition, we investigated an extension of our representation, by combining it with the state-of-the-art Fisher pooling representation [Lev et al., 2015]. In most collections the combined representation performed almost as well as the best of the two representations (hyperwords and Fisher pooling) considered separately.

1.3 Main Contributions

The main contributions that result from this dissertation are outlined below:

- A strategy capable of incorporating semantic information into text representations by clustering words with high similarity (i.e. hyperwords), aiming to reduce the inherent noise related to word embeddings in text classification.
- The formulation of a TF-IDF measure, based on the traditional one used for single words, but designed specifically for hyperwords.
- A robust representation originated from the combination of our strategy with a state-of-the-art pooling method.
- A thorough comparison between our method and other state-of-the-art strategies on a large set of datasets, performed with statistical rigor.
- A study comparing the discriminative power of hyperwords when compared to simply words.

1.4 Outline

This dissertation is organized as follows: Chapters 2 and 3 present background knowledge that are useful for contextualizing with the problems addressed in this dissertation, as well as recent relevant related work. Chapter 4 presents our proposed method, as well as some proposed improvements over it. Chapter 5 presents experiments demonstrating the performance of our method when compared to the Bag-of-Words and other strategies in the literature. It also supports some hypotheses that attempt to explain the superiority of hyperwords when compared to simple words. Chapter 6 presents conclusions and future work.

Chapter 2

Background

There are many ways of representing words and documents. In this chapter, different ways of representing both words and documents are presented and discussed in detail.

2.1 Vector Space Models

In vector space models, each document in a collection is represented as a vector in space. This representation allows easily comparing any two given documents by similarity (for instance, by computing the cosine similarity between two document vectors). It also provides a natural representation for feature-based machine learning models, with each dimension in the vector space being mapped to a distinct feature.

The two vector space model variants that we cover in this dissertation are the Bag-of-Words model and the TF-IDF model.

2.2 Bag-of-Words

In the *Bag-of-Words* model, words are represented using the "one-hot" encoding. This representation can be constructed as follows:

Definition 1. *Let $\mathbb{D}$ be a collection of documents and $\mathbb{V}$ the set of all words that occur in that collection. Each word is represented as a vector $\mathbf{x} \in \mathbb{R}^{|\mathbb{V}|}$, constructed as follows:*

1. Consider each word $w_i \in \mathbb{V}$. Let $\mathbf{x}^{(w_i)}$ be the vector corresponding to word w_i.
2. Make $\mathbf{x}^{(w_i)} = (x_1, x_2, ..., x_j, ..., x_{|\mathbb{V}|})_{j:|\mathbb{V}|}$ where $x_j = 1$ if $j = i$ and 0 otherwise.

In practice, each word w_i is represented as a vector with dimensionality $|\mathbb{V}|$, in which only one of the components is set to 1 (the one that corresponds to w_i) and

every other component is set to 0. A direct consequence of this representation is that words are linearly independent from each other.

In the "one-hot" encoding bag-of-words, also known as binary bag-of-words, each document is represented as a binary vector of dimensionality $|\mathbb{V}|$. For each word that occurs in document d, the weight 1 is assigned. For every word that does not occur in d, the weight 0 is assigned. Formally,

Definition 2. *Let $d \in \mathbb{D}$ be a document of some collection $\mathbb{D}$, and $\mathbf{x}^{(w_i)} \in \mathbb{R}^{|\mathbb{V}|}$ be as defined in Definition 1. Then the Bag-of-Words vector that represents d is defined as follows:*

$$\mathbf{d}_{\text{bow}} = \sum_{w_i \in d} \mathbf{x}^{(w_i)}$$

2.3 TF-IDF

Some weighting scheme can be applied in order to enhance the *Bag-of-Words* representation. The *TF-IDF* representation is a good example of weighting over the *Bag-of-Words* representation.

The TF-IDF weighting scheme breaks down into its two main components: namely, the TF (*term frequency*) and the IDF (*inverse document frequency*) components.

Term Frequencies aim to capture the "intensity" of some word in a document by counting how many times that word occur in that document. In general, the more frequent the word in a document, the stronger the chances that that document is related to that word. Therefore, $\text{tf}(w, d)$ equals how many times w occurs in d. It is important to note, however, that this measure is not sensitive to the expected frequency of the words in a collection. For instance, the words "the", "this", "a" are expected to occur more frequently in documents than the words "computer", "revolution" or "sports".

Document Frequencies aim to distinguish words that are too frequent from words that are less frequent in a collection. Common words like "the", "this" are expected to have greater document frequencies when compared to less common words like "computer" and "biology". Formally, for some collection $\mathbb{D}$,

$$\text{df}(w) = \frac{\alpha + \sum_{d \in \mathbb{D}} \text{I}(w \in d)}{|\mathbb{D}|}$$

where $\text{I}(x)$ is the indicator function that evaluates to 1 if x holds, 0 otherwise; and α is the smoothing component, usually set to 1.

The Inverse Document Frequency is usually defined as $\text{idf}(w_i) = \log\left(\frac{1}{\text{df}(w_i)}\right)$. It works by simultaneously promoting words that are less likely to occur in a collection (and tend to be the most informative) from words that occur too much (and tend to be less informative). Combined, the TF and IDF schemes work as a great weighting scheme over the Bag-of-Words representation, since it captures both local (TF) and global (IDF) word occurrence statistics of the collection. Formally,

Definition 3. *Let $d \in \mathbb{D}$ be a document of some collection $\mathbb{D}$, and $\mathbf{x}^{(w_i)} \in \mathbb{R}^{|\mathbb{V}|}$ be as defined in Definition 1. Then the TF-IDF vector that represents d is defined as follows:*

$$\mathbf{d}_{\text{tfidf}} = \sum_{w_i \in d} \mathbf{x}^{(w_i)} \text{tf}(w_i, d) \text{idf}(w_i)$$

2.4 Word Embeddings

In this representation, each word is represented as a continuous (multi-dimensional, rational) vector. Words are no longer independent from one another. In fact, it is possible to measure the similarity between two words using some measure (for instance, cosine similarity, inner product).

Word2Vec learns continuous word embeddings from plain text in an unsupervised way. In Word2Vec, a neural network is trained with streams of words and their respective contexts. The network is trained to maximize the probability of observing word w_n given words $(w_1, ..., w_{n-1})$ – CBOW – or the other way around – Skip-gram. The output is a matrix of word vectors or context vectors respectively. Since the neural network used has a simple linear hidden layer, the intensity of correlation between words is directly measured by the inner product between word embeddings. To reduce computational costs, Word2Vec uses fast training algorithms: hierarchical softmax and negative sampling [Goldberg and Levy, 2014]. Word2Vec is a popular choice for building word vectors due to their efficiency and simplicity. There are several implementations available, including the original C utility[1] and a Python library: gensim[2]. Word embeddings are illustrated in the example provided in figure 3.1.

[1] https://code.google.com/p/word2vec/
[2] https://radimrehurek.com/gensim/models/word2vec.html

Chapter 3

Related Work

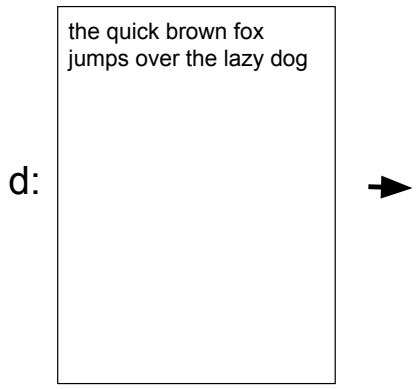
There is a vast and increasing amount of work related to exploiting word embeddings for effective Automated Document Classification. In this chapter, we cover the main topics and the most relevant and recent contributions to those topics.

3.1 Pooling-based Methods

Some proposals for representing documents based on their words' embeddings consist of aggregating the individual embeddings according to some strategy (a process we refer to as pooling), which results in a document representation that is directly derived from them. We discuss some of these proposals in this section.

3.1.1 Word Vector Centroids

One way of exploiting word embeddings for document representation is to create a centroid vector computed as the average of the vectors of all words belonging to a document, presenting the same dimensionality as the vectors of the words that compose the document (illustrated in figure 3.1). This approach has been evaluated for several tasks [Palakodety and Callan, 2014; Zhang et al., 2015; Xing et al., 2015; Amiri and III, 2016]. Although the centroid-based approaches are intuitive, much of the information is lost when vectors of words that compose a document are averaged together. In light of the limitations faced by the centroid-based approaches, other authors have proposed more robust techniques to extract information from word vectors and apply them to document representation.



V("the")=	(0.03, 0.11, ..., -0.15)	+
V("quick")=	(0.42, -0.32, ..., -0.07)	+
V("brown")=	(...)	+
V("fox")=	(...)	+
V("jumps")=	(...)	+
V("over")=	(...)	+
V("the")=	(...)	+
V("lazy")=	(...)	+
V("dog")=	(0.21, 0.41, ..., -0.17)	=
V(d)=	(-0.16, 0.05, ..., -0.15)	

Figure 3.1: Illustration of the generation of a Centroid vector from word embeddings. V(w) denotes the word embedding for w.

3.1.2 Fisher Vectors

In contrast to the simple centroid-based pooling approach, more sophisticated pooling strategies for Word2Vec representations were recently proposed [Lev et al., 2015]. The proposal defends that a proper pooling technique of the vectors of the text words may lead to at least very competitive results. This assumption is based on the benefits of pooling methods as one of the primary steps in many computer vision domains in the era before the advent of Deep Learning. The authors evaluated the use of principal component analysis (PCA) [Sanchez et al., 2013] or independent component analysis (ICA) [Hyvärinen and Oja, 2000] as an unsupervised pre-processing step that transforms the semantic vector space into independent semantic channels. After the pre-processing step, the authors propose the use of Fisher Vectors [Perronnin and Dance, 2007] as a pooling strategy for word embeddings, since it is today the leading pooling technique in computer vision domain and provides state-of-the-art results on many applications [Ken Chatfield and Zisserman, 2011; Peng et al., 2014]. As Lev et al. [2015] describes, the Fisher Vector is defined as the gradients of the log-likelihood of a multiset of vectors (in our case, the word embeddings $\mathbb{W}$) with respect to a Gaussian Mixture Model. Lev et al. [2015] observes that, empirically, the accuracy of the Fisher Vectors considering a mixture model of a single Gaussian is similar to that obtained from a mixture of multiple Gaussian distributions. The Fisher Vector of a single

Gaussian model with parameters $\lambda = \{\mu, \sigma\}$ is described as follows:

$$\frac{\partial \mathcal{L}(\mathbb{W}|\lambda)}{\partial \mu_i} = \sum_{\mathbf{w} \in \mathbb{W}} \frac{\mathbf{w}_i - \mu_i}{\sigma_i^2} \; ; \; F_{\mu_i} = \frac{|\mathbb{W}|}{\sigma_i^2} \tag{3.1}$$

$$\frac{\partial \mathcal{L}(\mathbb{W}|\lambda)}{\partial \sigma_i} = -\frac{|\mathbb{W}|}{\sigma_i} + \sum_{\mathbf{w} \in \mathbb{W}} \frac{(\mathbf{w}_i - \mu_i)^2}{\sigma_i^3} \; ; \; F_{\sigma_i} = \frac{2|\mathbb{W}|}{\sigma_i^2} \tag{3.2}$$

Each word is then represented as a vector that results from concatenating these gradients, normalized by F_{μ_i} and F_{σ_i}. Here, i ranges from 1 to D, in which D denotes the dimensionality of the word vectors used to generate the Fisher Vectors. For instance, for the pre-trained Word2Vec word embeddings, $D = 300$. Therefore, taking $\|$ as the concatenation operator:

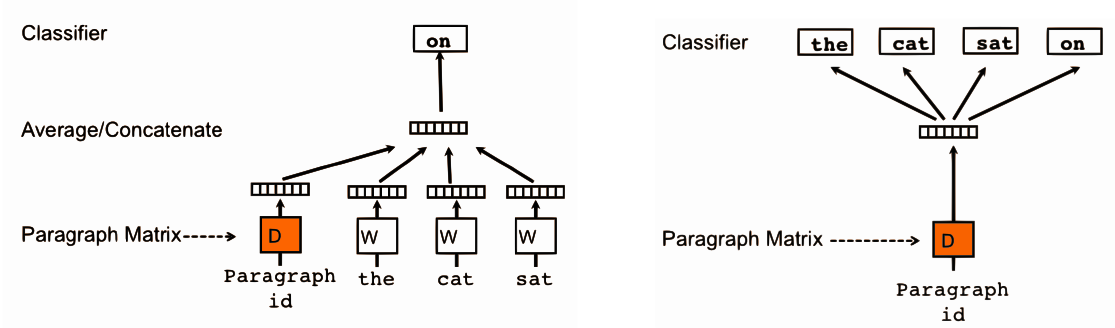
$$\text{FV}(\mathbf{w}) = \Big\|_{i=1}^{D} \left(\sqrt{F_{\mu_i}} \cdot \frac{\partial \mathcal{L}(\mathbf{w}|\lambda)}{\partial \mu_i} \right) \; \| \; \Big\|_{i=1}^{D} \left(\sqrt{F_{\sigma_i}} \cdot \frac{\partial \mathcal{L}(\mathbf{w}|\lambda)}{\partial \sigma_i} \right) \tag{3.3}$$

3.2 Document Vectors

In contrast to pooling techniques, Le and Mikolov [2014] propose Paragraph2Vec, an extension to the neural network model that represents words [Mikolov et al., 2013b] to arbitrary length sentences or documents. The main intuition behind Paragraph2Vec is the construction of dense document vectors that are trained to predict words for each document. Thus, each vector is a fixed-length representation of variable-length pieces of text. The method proposes two different approaches for training document vectors: the PV-DM and the PV-DBOW.

3.2.1 PV-DM

In PV-DM, a neural network softmax classifier is trained to maximize the probability of observing a word given its context. The algorithm learns word vectors (W) and paragraph vectors (D) using back-propagation. As a final step, it then updates the paragraph vectors (D), while keeping the word vectors (W) fixed, by back-propagating the errors only to the weight matrix D. This process can be seen as training the paragraph vectors to "memorize" what is missing from the current context in the paragraph (hence it is referred to by the authors as a distributed memory model). Figure 3.2a illustrates the network for generating D using the PV-DM model.



(a) Illustration of the PV-DM model. (b) Illustration of the PV-DBOW model.

Figure 3.2: Illustration of the Paragraph2Vec network models.
Source: Le and Mikolov [2014]

3.2.2 PV-DBOW

In PV-DBOW, paragraph vectors are trained to predict words randomly sampled from windows in the paragraph, regardless of the words' order in that paragraph. With respect to the results reported in the original Paragraph2Vec paper [Le and Mikolov, 2014], Mesnil et al. [2014] later observed that lower error rates were due to train and test data not being shuffled. Figure 3.2b illustrates the network for generating D using the PV-DBOW model.

3.3 "Task-aware" Word Embeddings

One limitation of the Word2Vec model is that it does not exploit information of the task at hand. For instance, assume one wants to generate and use word vectors for the task of text classification. The generated word vectors do not take into account the relations between documents and their labels, and therefore contain no information specific to the task at hand. One recent embedding model (Predictive Text Embeddings, or PTE) [Tang et al., 2015] has been proposed that builds word vectors taking into account the specific task they are built for. That is done by encoding a heterogeneous network of words, documents and labels in a low-dimensional space. The word's encoding can then be used as an embedding for that word. The authors also claim state-of-the-art performance when representing documents as the sum of the word vectors that compose some given document.

This model works by creating three bipartite graphs: a Word-Word network, a Word-Document network and a Word-Label network. Tang et al. [2015] describes them as follows:

Word-Word Network: A Word-Word co-occurrence network that captures word co-occurrence information in local contexts. It is a graph $G_{ww} = (V, E_{ww})$, in which the weight w_{ij} of edge between word v_i and v_j is given by the number of times that the two words co-occur in windows of a fixed size.

Word-Document Network: A graph $G_{wd} = (V \cup D, E_{wd})$ for a set of words V and documents D. E_{wd} is the set of edges between words and documents. The weight w_{ij} of the edge between word v_i and document d_j is given by the number of times v_i appears in d_j.

Word-Label Network: A graph $G_{wl} = (V \cup L, E_{wl})$ for a set of words V and labels L. E_{wl} is the set of edges between words and labels. The weight w_{ij} of the edge between word v_i and label l_j is defined as: $w_{ij} = \sum_{(d:l_d=j)} n_{di}$, where n_{di} is the term frequency of word v_i in document d, and l_d is the class label of document d.

The Predictive Text Embeddings model considers an heterogeneous network consisting of G_{ww}, G_{wd} and G_{wl} to create the low-dimensional representation vectors. It does so by generating embeddings for the vertices of this heterogeneous network. These embeddings ($\vec{u}_w$, $\vec{u}_d$, $\vec{u}_l$) are trained using stochastic gradient descent for optimizing the following function:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl} \tag{3.4}$$

In which each of the partial optimization functions (O_{ww}, O_{wd} and O_{wl}) can be computed individually by minimizing the following function for each network $G = (V_A, V_B, E)$:

$$\begin{aligned} O_{AB} &= \sum_{j \in B} \lambda_j d(\hat{p}(\cdot|v_j), p(\cdot|v_j)) \propto -\sum_{(i,j) \in E} w_{ij} \log p(v_j|v_i) \\ \text{of which, } p(v_j|v_i) &= \frac{\exp(\vec{u}_j^T \cdot \vec{u}_i)}{\sum_{j' \in B} \exp(\vec{u}_{j'}^T \cdot \vec{u}_i)} \end{aligned} \tag{3.5}$$

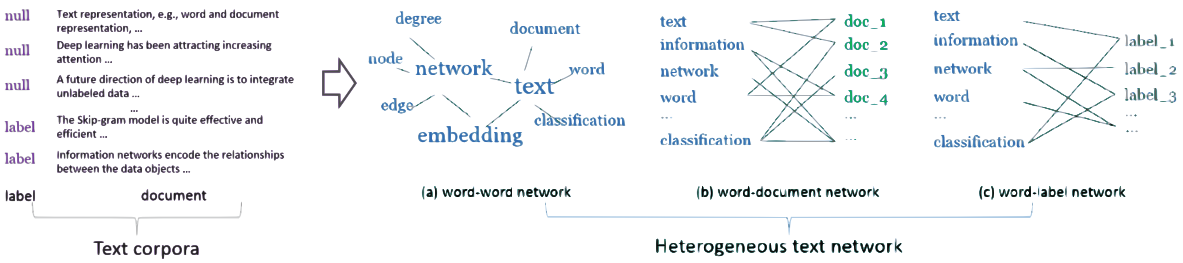


Figure 3.3: Illustration of the generation of heterogeneous networks for the PTE model.
Source: Tang et al. [2015]

In this dissertation, we use, as a baseline, the vectors obtained when minimizing O_{pte} with word vectors shared between all networks, which are optimized jointly. This is proposed by Tang et al. [2015] as the *joint* model. Documents are represented, following the proposal in the original work, as $\vec{d} = \frac{1}{n}\sum_{i=1}^{n}\vec{u}_i$, in which d is the document composed by words $(w_1, w_2, ..., w_n)$.

Chapter 4

Bag-of-Hyperwords: Using Word Embeddings for Improved Text Representation

In this chapter, we present our strategy of exploiting word similarity information related to the terms present in the documents to be used in a way that is compatible with the TF-IDF document representation.

The similarity context is obtained by means of word embeddings, that can be obtained either from a precomputed set (such as the Google News dataset[1]) or by running Word2Vec, PTE, or other method on the collection to obtain the word embeddings. In this dissertation, we use pre-trained word embeddings obtained using Word2Vec as a proxy to word similarity. We choose to do so because it is computationally cheaper and provides better reproducibility.

We propose to use word similarities to improve the text representation in a two-step approach:

1. For each term t in the training set, we compute a vector we refer to as *hyperword*. The hyperword vector for term t contains, for each term t' in the training set, the similarity value between t' and t. Small similarity values are generally truncated. If using Word2Vec for word similarity, the hyperword for term t contains terms of the vocabulary that are most semantically related to t. This relies on the premise that Word2Vec embeddings similarities are correlated with semantic relatedness.

2. Then, we compute a modified version of TF-IDF weighting for the new features (hyperwords). These new features can then be used to represent the document

[1] https://code.google.com/p/word2vec/

in place of the original BoW features, resulting in a "*Bag-of-Hyperwords*" representation.

4.1 Hyperword generation

Let $\mathbb{V}$ be the vocabulary of terms present in the set of training documents $\mathbb{D}$. Also, let $\mathbb{W}$ be the set of word embeddings representing each term in $\mathbb{V}$. Thus, each term t in $\mathbb{V}$ should have a corresponding vector in $\mathbb{W}$ [2].

We define hyperword h_t as a vector with dimensionality $|\mathbb{V}|$, such that $h_t = (w_1, w_2, \cdots, w_{|\mathbb{V}|}) : w_i \in [0,1], 1 \leq i \leq |\mathbb{V}|$. Initially, we compute the similarity $\omega(u,v)$ between every pair of vectors u and v of $\mathbb{W}$. We use the cosine similarity measure to compute $\omega(u,v)$, as defined in Eq. 4.1. Note that other distance measures could be applied as well (e.g., euclidean distance). We use cosine similarity as a proof-of-concept due to its simplicity.

$$\omega(u,v) = \frac{\sum_i^l u_i \cdot v_i}{\sqrt{\sum_i^l u_i^2} \cdot \sqrt{\sum_i^l v_i^2}} \tag{4.1}$$

Next, we create a hyperword h_t for each term $t \in \mathbb{V}$ according to Eq 4.2.

$$h_t = (w_1, w_2, \cdots, w_{|\mathbb{V}|}) : w_i = \begin{cases} \omega(t,t_i) & \text{if } \omega(t,t_i) \geq \alpha \\ 0 & \text{otherwise} \end{cases}, 1 \leq i \leq |\mathbb{V}|. \tag{4.2}$$

where α is a similarity threshold which controls the inclusion of the value of the similarity between a term t_i and term t in the hyperword h_t. In other words, Eq. 4.2 means that the hyperword h_t has a component corresponding to each term t_i in $\mathbb{V}$. The weight w_i for the component corresponding to t_i is equal to the cosine similarity between the vectors for t and t_i in the word vector space if this similarity is greater than or equal to a threshold α. Otherwise, the weight w_i is equal to zero.

Since α is used as threshold for a cosine similarity value, its effective domain is contained within the interval $[-1,1]$. However, we are only interested in terms with positive cosine similarity, since $0 \leq w_t \leq 1 \forall t$. Therefore, the domain of α is reduced to the range $[0,1]$. When $\alpha = 0$ the similarities of every term in $\mathbb{V}_T$, as long as non-negative, are included in h_t. Otherwise, if $\alpha = 1$ only the similarity of t to itself (i.e. $\omega(t,t) = 1.0$) is included in h_t. The appropriate selection of a value for parameter α is an important aspect of generating good hyperwords, and alpha can be seen as

[2] We deal with out-of-vocabulary (OOV) words by setting $h_t = (1 * I[t_i = t])_{1 \leq i \leq |\mathbb{V}|}$, where $I[\cdot]$ denotes the indicator function. Put another way, for out of vocabulary words, their semantic context is composed solely by the word itself, with weight equal to 1.

a parameter that controls the degree of specificity of the hyperwords. The lower the alpha, the more generic / less specific is the hyperword. As alpha increases to values closer to one, the Bag-of-Hyperwords model becomes increasingly more similar to the traditional Bag-of-Words model.

After the above described process is finished, the resulting set $\mathcal{H} = \{h_t : t \in \mathbb{V}\}$ contains the final set of hyperwords that can be used as new features for document representation. In order to represent a document d as a bag of hyperwords, we need to compute TF-IDF weights for each hyperword to be inserted in d. We describe this weighting scheme in Section 4.2.

4.2 TF-IDF weights for hyperwords

Having computed a set of hyperwords, we now define the TF-IDF-based weighting scheme for these new features. Recall from Section 2.3 that TF is a document-level statistic that measures the importance of terms in their documents, while the IDF is a collection-level statistic that measures how informative each term is for the collection. We aim to redefine the TF and IDF statistics for hyperwords without changing their semantics. That is, we want the TF of a hyperword to reflect its importance in a document, and the IDF of a hyperword to reflect its importance in the collection. Since hyperwords are created based on semantic similarity of terms, the conventional TF-IDF value over the terms is not capable of weighting these features while taking full advantage of the information provided by them, as none of its components take the similarity between words into account. Our motivation is to combine the two aspects of the conventional TF-IDF value with the semantic information of a hyperword. In what follows, we propose a modified version of the TF-IDF score for hyperwords. That score can be used to compute appropriate weights for hyperwords to be used in the representation of document d. The TF-IDF for hyperword h_t in document d (tf-idf(h_t, d)) is defined according to Eq. 4.3, with tf(h_t, d) and idf(h_t) being redefined in the following section.

$$\text{tf-idf}(h_t, d) = \text{tf}(h_t, d)\,\text{idf}(h_t) \tag{4.3}$$

4.2.1 Term Frequency of a Hyperword

First, we define how we compute the TF part of the measure (tf(h_t, d)) of hyperword h_t in document d as described in Eq. 4.4.

$$\mathrm{tf}(h_t, d) = \sum_{t_i \in h_t} \mathrm{tf}(t_i, d) w_i \tag{4.4}$$

The value of $\mathrm{tf}(h_t, d)$ corresponds to the sum of the products of the term frequencies $\mathrm{tf}(t_i, d)$ of each term t_i occurring in document d by the weight w_i of term t_i in hyperword h_t. Remember that the value of w_i in vector h_t is computed in Eq. 4.2.

The main intuition behind Eq. 4.2 is that each term $t_i \in d$ should contribute to the score assigned to h_t. That contribution is weighted by the similarity between t and t_i, given by w_i. Note that, since w_i is used instead of directly applying the cosine similarity $\omega(t, t_i)$, the minimum required similarity for a term to contribute with a hyperword's score can be controlled by adjusting α.

4.2.2 Inverse Document Frequency of a Hyperword

To compute the IDF of hyperword h_t we first define the vocabulary $\mathbb{V}_{d,h_t}$ composed by all terms in document d which have the weight (w_i) not equal to zero in hyperword h_t. This is formally defined in Eq. 4.5.

$$\mathbb{V}_{d,h_t} = \{t_i \in d | w_i > 0 \text{ in } h_t\} \tag{4.5}$$

Next, we compute the mean of the values of the weights in hyperword h_t of terms occurring in vocabulary $\mathbb{V}_{d,h_t}$, according to Eq. 4.6.

$$\mu_{h_t,d} = \frac{1}{|\mathbb{V}_{d,h_t}|} \sum_{t_i \in \mathbb{V}_{d,h_t}} w_i \tag{4.6}$$

Finally we compute IDF hyperword h_t as defined in Eq 4.7, where $\mathbb{D}$ is the training set.

$$\mathrm{idf}_{h_t} = \log\left(\frac{|\mathbb{D}|}{\sum_{d \in \mathbb{D}} \mu_{h_t,d}}\right) \tag{4.7}$$

In order to better understand the meaning of idf_{h_t}, let's turn to the traditional IDF measure. As we have previously indicated, when $\alpha = 1.0$, our representation is identical to the traditional BoW. For that, $\mathbb{V}_{d,h_t} = \{t\}$ if $t \in d$, $\emptyset$ otherwise; which implies that $\mu_{h_t,d} = 1.0$ if $t \in d$, 0 otherwise. It follows that, when $\alpha = 1.0$, $\sum_{1 \leq d \leq |\mathbb{D}|} \mu_{h_t,d}$ is exactly the same as the document frequency for term t.

For $\alpha < 1$, $\mu_{h_t,d}$ is the simple average between the similarities of terms in h_t whose w_i is greater than 0. Its value is comprised between 0 and 1. Algorithms 1 and 2 give an overall perspective of hyperwords generation and weighting schemes.

Algorithm 1 Bag-of-Hyperwords Generation

function ComputeBagOfHyperwords($\mathbb{V}, \mathbb{W}, \mathbb{D}, \alpha$)
  $M \leftarrow \mathbf{0}$ ▷ 0-initialized $|\mathbb{D}| \times |\mathbb{V}|$ matrix
  $\mathcal{H} \leftarrow$ GenerateHyperwords($\mathbb{V}, \mathbb{W}, \alpha$)
  for $d_i \in \mathbb{D}$ **do**
    for $h_j \in \mathcal{H}$ **do**
      $M[i,j] \leftarrow \mathrm{TF}(h_j, d_i) * \mathrm{IDF}(h_j, \mathbb{D})$
  return M
function GenerateHyperwords($\mathbb{V}, \mathbb{W}, \alpha$)
  $\mathcal{H} \leftarrow \{\}$ ▷ The set of hyperwords to be returned
  for $w_i \in \mathbb{V}$ **do**
    $h_i \leftarrow (0, 0, ..., 0)_{|\mathbb{V}|}$
    $h_i[i] \leftarrow 1$
    for $w_j \in \mathbb{V}$ **do**
      if $w_i, w_j \in \mathbb{W}$ **and** $\omega(w_i, w_j) \geq \alpha$ **then**
        $h_i[j] \leftarrow \omega(w_i, w_j)$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{h_i\}$
  return $\mathcal{H}$

Algorithm 2 Bag-of-Hyperwords Statistics

function $\mathrm{TF}(h_j, d)$
  $\mathrm{tf}_{h_j,d} \leftarrow 0$
  for $t_i \in d$ **do**
    $\mathrm{tf}_{h_j,d} \leftarrow \mathrm{tf}_{h_j,d} + h_j[i] * \mathrm{tf}(t_i, d)$ ▷ $\mathrm{tf}(t_i, d)$ refers to the original counts of t_i in d
  return $\mathrm{tf}_{h_j,d}$
function $\mathrm{IDF}(h_j, \mathbb{D})$
  $\mathbf{df}_{h_j} \leftarrow 0$
  for $d \in \mathbb{D}$ **do**
    $\mathbb{V}_{d,h_j} \leftarrow \{\}$
    for $t_i \in d$ **do**
      if $h_j[i] > 0$ **then**
        $\mathbb{V}_{d,h_j} \leftarrow \mathbb{V}_{d,h_j} \cup \{t_i\}$
    $\mu_{h_j,d} \leftarrow \frac{1}{|\mathbb{V}_{d,h_j}|} \sum_{t_i \in \mathbb{V}_{d,h_j}} h_j[i]$
    $\mathbf{df}_{h_j} \leftarrow \mathbf{df}_{h_j} + \mu_{h_j,d}$
  return $\log\left(\frac{|\mathbb{D}|}{\mathbf{df}_{h_j}}\right)$

4.3 Dynamic Alpha Selection: Increasing Hyperwords' Discriminative Power

We have described the overall process of generating Hyperwords and computing their TF-IDF values. A limitation of the model we have explained so far is that all Hy-

perwords are generated using the same value for the similarity threshold (α). If we interpret Hyperwords as representing concepts of words that are clustered together (such as the concept formed by the joining words "car" and "bus"), then a single value of α, shared between all Hyperwords, imposes that the degree of generality/specificity of the concepts being captured is the same for every Hyperword.

Motivated by adjusting the degree of specificity of each Hyperword to the one that best suits the collection of documents, we propose a method for automatically picking a value of α for each Hyperword, named Dynamic Alpha Selection. This method finds, for each Hyperword, the value of α that makes it the most discriminative possible. Consider the following toy example:

$\mathbb{D}$ = { d_1:"nice bus", d_2:"nice car", d_3:"bad bus", d_4:"bad rain"} category(d_1) = 1 category(d_2) = 1 category(d_3) = 2 category(d_4) = 3	**Similarities:** ω(nice, bad) = 0.6 ω(bus, car) = 0.6 ω(nice, bus) = 0.3 ω(nice, car) = 0.3 ω(bad, bus) = 0.3 ω(bad, car) = 0.3 ω(nice, rain) = 0.1 ω(bad, rain) = 0.1 ω(bus, rain) = 0.1 ω(car, rain) = 0.1	$\mathbb{V}$ = {nice, bad, bus, car, rain} $\mathcal{H}$ = {h_{nice}, h_{bad}, h_{bus}, h_{car}, h_{rain}}

A few details to notice about this example: words "bus" and "car" co-occur inside the same category (category 1 for d_1 and d_2). Meanwhile, words "nice" and "bad" never occur in the same category[3]. For this specific collection, it is beneficial to group "bus" and "car" together, but it is not to group "nice" and "bad". This is impossible to do, if Hyperwords are generated with a global alpha, for the given similarity values. This is illustrated below.

Now, suppose that the documents are clustered through a recursive partition of the feature space over features h_{nice} and h_{car}. Figure 4.1 illustrates how the discriminative power of Hyperwords change for different choices of α.

We now describe the process of generating the best values of α such that each Hyperword has maximum discriminative power. In order to assess the discriminative power of a Hyperword, we use the Mutual Information, or Information Gain metric (that is described in detail in section 5.4.2). Formally, let $h_t^{(\alpha')}$ denote the Hyperword h_t built for term t according to equation 4.2, considering $\alpha = \alpha'$. Then each Hyperword

[3]Keep in mind that categories usually carry some real world meaning (e.g., topics in a magazine, such as "sports", "entertainment"), but in this toy example they are arbitrarily set for illustration purposes.

$0.3 < \alpha \leq 0.6$	h_{nice}={"nice": 1.0; "bad": 0.6} h_{bad}={"bad": 1.0; "nice": 0.6} h_{bus}={"bus": 1.0; "car": 0.6} h_{car}={"car": 1.0; "bus": 0.6} h_{rain}={"rain": 1.0}	d_1={h_{nice},h_{bad},h_{bus},h_{car}} d_2={h_{nice},h_{bad},h_{bus},h_{car}} d_3={h_{nice},h_{bad},h_{bus},h_{car}} d_4={h_{nice},h_{bad},h_{rain}}
$\alpha > 0.6$	h_{nice}={"nice": 1.0} h_{bad}={"bad": 1.0} h_{bus}={"bus": 1.0} h_{car}={"car": 1.0} h_{rain}={"rain": 1.0}	d_1={h_{nice},h_{bus}} d_2={h_{nice},h_{car}} d_3={h_{bad},h_{bus}} d_4={h_{bad},h_{rain}}
Ideal dynamic α	h_{nice}={"nice": 1.0} h_{bad}={"bad": 1.0} h_{bus}={"bus": 1.0; "car": 0.6} h_{car}={"car": 1.0; "bus": 0.6} h_{rain}={"rain": 1.0}	d_1={h_{nice},h_{bus},h_{car}} d_2={h_{nice},h_{bus},h_{car}} d_3={h_{bad},h_{bus},h_{car}} d_4={h_{bad},h_{rain}}

can be built according to the following equation:

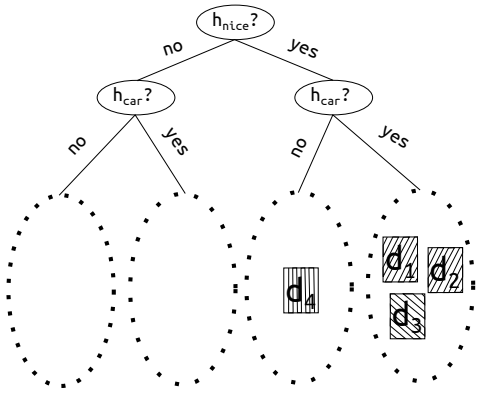
$$h_t^{(\alpha^*)} \ : \ \alpha^* \ = \ \arg\max_{\alpha'} \left(\text{MI}(Y; h_t^{(\alpha')}) \right) \tag{4.8}$$

Since Hyperwords are discrete sets, one can simply evaluate the objective function for each possible value of alpha. That is, the values of alpha such that $|h_{t,k+1}^{(\alpha')}| = |h_{t,k}^{(\alpha')}| - 1$ (considering that the $|h_t|$ operator corresponds to the number of words with weight greater than 0 in hyperword h_t). Another (sub-optimal) way is to evaluate values of α' from a predetermined set, such as $\{0.3, 0.4, ..., 1.0\}$.

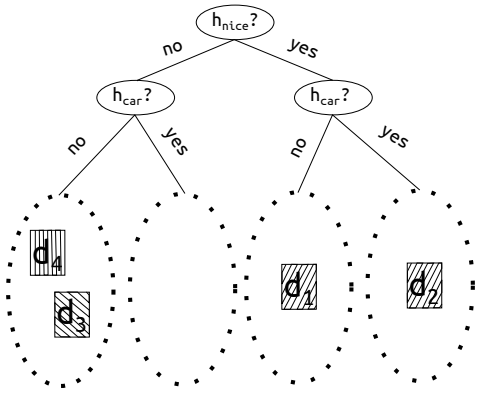
4.4 Merging Redundant Hyperwords

Note that once we select an appropriate value for α, each hyperword h_t keeps the values of similarities of the terms most similar to t according to the semantics established by the word embeddings. However, it is possible that some hyperwords are quite similar to others. Furthermore, there may be hyperwords with only one non-zero word component. All these redundant information present two problems when using hyperwords as features: (i) the growth in complexity due to the unnecessarily high dimensionality, and (ii) introduction of useless or noisy information.

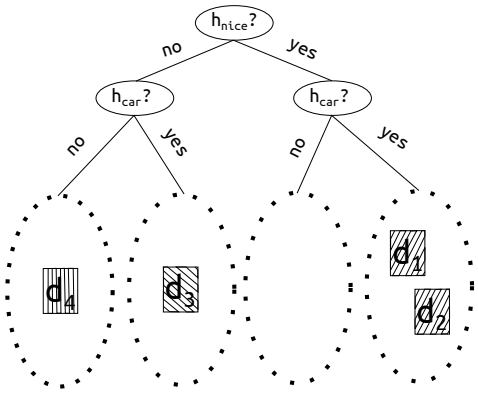
To cope with the two problems, we introduce a second step in the process of obtaining hyperwords which corresponds to grouping together similar hyperwords,



(a) Partition when $0.3 < \alpha \leq 0.6$. This partition is not the most discriminative, as d_3 is of a different category than d_1 and d_2.



(b) Partition when $\alpha > 0.6$. This partition is not the most discriminative, as d_3 is of a different category than d_4.



(c) Partition when α is chosen dynamically for each Hyperword. This is the most discriminative setting, as every region is pure (contains only documents of the same category).

Figure 4.1: Choosing the best α for increased discriminative power of individual Hyperwords.

aiming to reduce the set of hyperwords to be included in the final document representation. In this process, we once again use the cosine similarity measure, this time for comparing hyperwords against each other. This clustering process is a recursive procedure computed as described in the following steps:

Step 1 Initialize $\mathcal{H}$ with the set of all h_t as defined in Eq. 4.2.

Step 2 $\mathcal{H}_0 = \mathcal{H}$

Step 3 Compute $\omega(h_i, h_j)$ for each pair of hyperwords $\langle h_i, h_j \rangle$, $h_i, h_j \in \mathcal{H}, i \neq j$.

Step 4 If $\omega(h_i, h_j) \geq 1 - \beta$, define $h_{i,j} = (\frac{w_{i_1}+w_{j_1}}{2}, \frac{w_{i_2}+w_{j_2}}{2}, \cdots \frac{w_{i_{|\mathcal{V}|}}+w_{j_{|\mathcal{V}|}}}{2})$, remove h_i and h_j from $\mathcal{H}$ and insert $h_{i,j}$ in $\mathcal{H}$.

Step 5 if $\mathcal{H} \neq \mathcal{H}_0$ go to step 2, else stop.

After the above described process is finished, the resulting set $\mathcal{H}$ contains the final set of hyperwords that should be used to represent the documents. The TF-IDF weights for these hyperwords can be obtained the same way as described in Section 4.2.

Chapter 5

Experimental Evaluation

In this chapter, we experimentally study the Bag-of-Hyperwords model on five topic datasets and nineteen sentiment analysis datasets from different contexts. In section 5.1, we present the experimental setup. In section 5.2 we provide an experimental evaluation of the Bag-of-Hyperwords model in comparison with the traditional Bag-of-Words model as well as some other baselines.

5.1 Experimental Setup

5.1.1 Datasets

In order to evaluate the proposed model, we consider five real-world topic datasets. We also consider nineteen sentiment datasets from scenarios such as product reviews, messages and comments in social media. For all datasets, we perform a traditional pre-processing task: we remove stop-words, using the standard SMART list, and apply a simple feature selection by removing terms with low "document frequency (DF)"[1]. Next, we give a brief description of the datasets:

4 Universities (4uni), a.k.a, WebKB contains Web pages collected from Computer Science departments of four universities by the Carnegie Mellon University (CMU) text learning group[2]. There is a total of $8,277$ web pages, classified in 7 categories (such as student, faculty, course and project web pages).

20 Newsgroups (20ng) contains $18,828$ newsgroup documents, partitioned almost evenly across 20 different newsgroups categories[3]. 20ng has become a popular

[1] We removed all terms that occur in less than three documents (i.e., DF<3).

[2] `http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/`

[3] `http://qwone.com/~jason/20Newsgroups/`

dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering.

ACM-DL (acm) a subset of the ACM Digital Library with 24,897 documents containing articles related to Computer Science. We considered only the first level of the taxonomy adopted by ACM, whereas each document is assigned to one of 11 classes.

Reuters (reut90) is a classical text dataset, composed by news articles collected and annotated by Carnegie Group, Inc. and Reuters, Ltd. We consider here a set of 13,327 articles, classified into 90 categories.

AG's News (agnews) is a subset of the AG's News corpus dataset, with 241,469 articles classified in 4 categories (world, sports, business and sci/tech) chosen in previous works [Lev et al., 2015].

Sentiment Datasets are nineteen publicly available sentiment benchmarks of product reviews, short messages and comments gathered from different works. They are named aisopos_tw [Fotis Aisopos, 2014], debate [Diakopoulos and Shamma, 2010], narr_tw [Narr et al., 2012], pappas_ted [Pappas and Popescu-Belis, 2013], pang_movie [Pang and Lee, 2004], sanders_tw[4], ss_bbc, ss_digg, ss_myspac, ss_rw, ss_twitter, ss_youtube [Thelwall, 2013], stanford_tw [Go et al., 2009], semeval_tw[5],vader_amzn, vader_movie, vader_nyt, vader_tw [Hutto and Gilbert, 2014] and yelp_review[6]. As proposed by Canuto et al. [2016], we consider only documents with a clear polarity (positive or negative).

Table 5.1 shows some characteristics of the datasets used to evaluate our model. The first column indicates the name of the dataset used in our experiments, the second column is the number of documents in the dataset, the third shows the number of features (words) represented in the dataset, the fourth corresponds to the average number of words per document (density), and the last column shows the number of classes.

5.1.2 Evaluation, Algorithms and Procedures

The effectiveness of our classification experiments was evaluated using two standard text categorization measures: The micro-averaged F_1 score (MicroF_1) and the macro-

[4] http://www.sananalytics.com/lab/twitter-sentiment
[5] https://www.cs.york.ac.uk/semeval-2013/task2
[6] http://www.yelp.com/dataset_challenge

	Dataset	#Doc	#Feat	Density	#Classes
Topic data	4uni	8,277	40,194	140.3	7
	20ng	18,828	61,049	130.8	20
	acm	24,897	59,990	38.8	11
	reut90	13,327	19,589	78.2	90
	agnews	241,469	57,257	25.5	4
Sentiment data	aisopos_tw	278	1,493	13.0	2
	debate	1,979	3,360	11.5	2
	narr_tw	1,227	3,508	11.3	2
	pappas_ted	727	1,635	11.7	2
	pang_movie	10,662	12,432	13.9	2
	sanders_tw	1,091	3,102	13.5	2
	ss_bbc	752	5,655	40.3	2
	ss_digg	782	4,015	22.2	2
	ss_myspace	834	2,639	14.7	2
	ss_rw	705	4,595	43.3	2
	ss_twitter	2,289	7,777	13.8	2
	ss_youtube	2,432	6,275	12.2	2
	stanford_tw	359	1,620	12.0	2
	semeval_tw	3,060	9,087	16.4	2
	vader_amzn	3,610	3,678	11.9	2
	vader_movie	10,568	11,980	14.0	2
	vader_nyt	4,946	8,756	13.0	2
	vader_tw	4,196	7,346	11.2	2
	yelp_review	5,000	19,398	71.5	2

Table 5.1: Dataset characteristics

averaged F_1 score (MacroF_1) [Lewis et al., 2004; Yang, 1999]. While the MicroF_1 measures the classification effectiveness over all decisions (i.e., the aggregated contingency tables of all classes), the MacroF_1 measures the classification effectiveness for each individual class and averages them. They are formally defined below:

5.1.2.1 MicroF_1 and MacroF_1

Let $\mathcal{Z}(d) \rightarrow y$ be a classification model that maps a document $d \in \mathbb{D}$ to a category $y \in \mathbb{C}$; $\mathbb{C}$ be the set of classes for a document collection $\mathbb{D}$. Let category(d) denote the category of document d according to the ground truth. The F_1-score is defined as follows:

$$F_1 \;=\; H(\text{precision}, \text{recall}) \;=\; 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.1}$$

In which the $H(\cdot)$ function denotes the harmonic mean. For the MicroF_1, the precision and recall are defined as follows:

$$\text{precision}_{\text{micro}} \;=\; \text{recall}_{\text{micro}} \;=\; \frac{\sum\limits_{d \in \mathbb{D}} I[\mathcal{Z}(d) = \text{category}(d)]}{|\mathbb{D}|} \tag{5.2}$$

In which I[predicate] is the Indicator Function (equals 1 if predicate is true, 0 otherwise). For the MacroF_1, the precision and recall are defined as follows:

$$\text{precision}_{\text{macro}} = \frac{1}{|\mathbb{C}|} \sum_{y \in \mathbb{C}} \left(\frac{1}{|\{d \in \mathbb{D} \ : \ \mathcal{Z}(d) = y\}|} \sum_{d \in \mathbb{D} \ : \ \mathcal{Z}(d) = y} I[y = \text{category}(d)] \right) \tag{5.3}$$

$$\text{recall}_{\text{macro}} = \frac{1}{|\mathbb{C}|} \sum_{y \in \mathbb{C}} \left(\frac{1}{|\{d \in \mathbb{D} \ : \ \text{category}(d) = y\}|} \sum_{d \in \mathbb{D} \ : \ \text{category}(d) = y} I[\mathcal{Z}(y) = y] \right) \tag{5.4}$$

Usually, when evaluating the performance of a classifier, the document collection $\mathbb{D}$ is split into train and test partitions. The classifier ($\mathcal{Z}(d) \rightarrow y$) is trained using $\mathbb{D}_{\text{train}}$, and the Micro$F_1$ and MacroF_1 are computed for $\mathbb{D}_{\text{test}}$.

5.1.2.2 Experimental Methodology

All experiments were executed using a 5-fold cross-validation procedure. The parameters were set via cross-validation on the training set, and the effectiveness of the algorithms running with distinct types of vector representation were measured in the test partition.

In order to evaluate the performance of different document representations, we use the SVM classifier using the LinearSVC class implemented in `scikit-learn` [Pedregosa et al., 2011]. For the textual documents, we also run the Random Forest classifier. We set the parameter number of trees to 200, since Breiman et al. [1984] recommends to use at least 100 trees. The remaining parameters are kept as the implementation defaults.

With exception of Bag-of-Words, all baselines for text representation depend on word embeddings. As in previous works based on word embeddings [Lev et al., 2015; Yuan et al., 2014; Le and Mikolov, 2014], all experiments involving the word vectors use the pre-computed word embeddings trained on the Google News Corpus[7]. We detail additional information about the description and parametrization of each approach as follows.

[7]The pre-computed word embeddings are available at `https://code.google.com/p/word2vec/`.

The traditional Bag-of-Words (**BoW**) representation in our experiments was weighted using the usual TF-IDF scheme, described in section 2.3.

The α parameter of the proposed hyperwords representation was chosen among six values varying from 0.4 to 0.9. The β parameter was chosen among five values varying from 0.3 to 0.7. The Dynamic Alpha Selection was chosen between enabled and disabled. Best values were determined with cross-validation within the training set.

We use two pooling strategies in our experiments: the mean pooling, which generates a centroid vector (**Centroid**) and the Fisher Vector pooling (**FisherP**). We use the mean pooling without any pre-processing step, since this is the most popular approach [Amiri and III, 2016]. For the Fisher pooling, we select the best space transforming approach (none, PCA or ICA) for each dataset as a pre-processing step, following suggestions proposed by Lev et al. [2015].

We evaluate the paragraph vector approach (**Paragraph2Vec**) proposed by Yuan et al. [2014] considering the best parameter configuration for each dataset. We tested different values for alpha (0.01, 0.025, 0.05 and 0.1), window size (5, 10, 15), initialization of the word embeddings (initializing with pre-processed word embeddings or random values) and dimension to represent the document (100, 200, 300 and 400). In all experiments, we use the combined approach of PV-DBOW and PV-DM as recommended by the authors.

The **PTE** vectors are obtained using the parameters as they were recommended by Tang et al. [2015]. We used the authors' implementation[8]. In order to include PTE as a baseline, we represent documents as the mean of the word vectors that compose a document, following the original proposal.

We assess the statistical significance of our results by means of a paired t-test with 95% confidence and Holm correction to account for multiple tests. We indicate the relative performance of our method as follows: a green upward triangle (▲) means our method is statistically superior to all other methods; a gray dot (•) means our method is statistically tied to the best method; and a red downward triangle (▼) means that another method is the best and is statistically superior to ours. This test assures that the best results, marked with a green triangle (▲), are statistically superior to others. Statistical ties are represented as a gray dot (•), while losses are represented as red downward triangles (▼). The obtained results (and their 95% confidence intervals) are described in Section 5.2.

[8] `https://github.com/mnqu/PTE`

5.2 Experimental Results

In this section, we discuss the performance of the Bag-of-Hyperwords model when compared to the Bag-of-Words, as well as baselines that propose alternative text representations relying on word embeddings.

Datasets	Hyperwords		BoW		Centroid	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
acm	74.0 (0.8) ▲	61.9 (1.8) ▲	68.8 (0.4)	56.5 (0.8)	63.4 (1.1)	49.7 (0.4)
20ng	91.1 (0.9) ▲	91.0 (0.9) ▲	89.5 (0.4)	89.4 (0.5)	77.5 (1.5)	76.3 (1.3)
4uni	75.6 (3.4) ▲	64.0 (2.6) ▲	69.8 (2.1)	54.2 (2.0)	70.7 (0.8)	56.5 (2.4)
reut90	68.9 (1.5) ▲	30.8 (3.6) ●	65.5 (1.0)	29.6 (2.9)	70.1 (0.5)	29.2 (3.2)
agnews	91.5 (0.1) ▲	90.9 (0.1) ▲	91.2 (0.1)	90.7 (0.1)	87.8 (0.2)	87.0 (0.2)

Table 5.2: Average $MicroF_1$ and $MacroF_1$ (and their 95% confidence interval) of different document representations (BoW, Centroid) on the SVM classifier for Topic Datasets.

Datasets	Hyperwords		Paragraph2vec		FisherP		PTE	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
acm	74.0 (0.8) ●	61.9 (1.8) ▲	63.4 (1.0)	45.4 (1.0)	67.5 (0.5)	54.2 (2.2)	74.3 (0.6)	60.6 (0.8)
20ng	91.1 (0.9) ▼	91.0 (0.9) ▼	81.5 (0.6)	80.5 (0.5)	80.4 (0.8)	79.5 (0.7)	96.5 (0.5)	96.4 (0.2)
4uni	75.6 (3.4) ●	64.0 (2.6) ●	66.8 (0.9)	52.0 (2.0)	78.8 (1.0)	66.3 (0.8)	72.0 (2.4)	57.3 (2.8)
reut90	68.9 (1.5) ●	30.8 (3.6) ●	69.2 (0.9)	27.0 (2.2)	70.1 (0.4)	28.1 (2.3)	68.0 (1.1)	27.6 (3.7)
agnews	91.5 (0.1) ▲	90.9 (0.1) ▲	84.8 (0.3)	83.6 (0.3)	88.4 (0.4)	87.5 (0.3)	77.8 (0.2)	77.1 (0.2)

Table 5.3: Average $MicroF_1$ and $MacroF_1$ (and their 95% confidence interval) of different document representations (P2V, FisherP, PTE) on the SVM classifier for Topic Datasets. Statistical tests include BoW and Centroid methods.

Algorithm	Winning Counts		Total
	MicroF_1	MacroF_1	
Hyperwords	4	4	8
PTE	3	3	6
FisherP	2	2	4
BoW	0	1	1
Centroid	0	1	1
Paragraph2Vec	1	0	1

Table 5.4: Number of times each algorithm was a top performer in the evaluation, using the SVM classifier, for a total of 5 topic datasets.

5.2.1 Topic Datasets

We start by presenting the effectiveness results obtained in the topic datasets. Tables 5.2, 5.3, 5.5 and 5.6 show the results of the traditional BoW, the proposed approach (Hyperwords) some of the most recent proposals to represent documents using word embeddings (Centroid, FisherP, Paragraph2Vec and PTE). The proposed approach was capable of achieving the best results (or at least statistically tying with the best) in

Datasets	Hyperwords		BoW		Centroid	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
acm	72.6 (0.5) ▲	58.0 (0.5) ▲	71.8 (0.9)	56.6 (0.9)	55.2 (0.9)	39.6 (0.7)
20ng	87.3 (0.8) ●	86.8 (1.2) ●	87.1 (0.9)	86.5 (0.9)	68.0 (1.1)	66.2 (0.9)
4uni	79.0 (1.1) ▲	66.5 (2.1) ▲	78.0 (0.9)	57.9 (2.2)	70.5 (1.1)	50.4 (2.7)
reut90	65.6 (1.0) ▲	27.7 (2.6) ▲	61.7 (0.6)	19.4 (2.4)	59.1 (1.0)	14.5 (1.2)
agnews	90.6 (0.1) ▲	89.9 (0.1) ▲	90.1 (0.1)	89.2 (0.1)	87.0 (0.2)	86.0 (0.3)

Table 5.5: Average $MicroF_1$ and $MacroF_1$ (and their 95% confidence interval) of different document representations (BoW, Centroid) on the RF classifier for Topic Datasets.

Datasets	Hyperwords		Paragraph2vec		FisherP		PTE	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
acm	72.6 (0.5) ▲	58.0 (0.5) ▲	57.1 (1.0)	39.8 (1.0)	53.9 (0.7)	37.4 (0.8)	70.0 (1.1)	55.5 (1.3)
20ng	87.3 (0.8) ▼	86.8 (1.2) ▼	70.8 (1.1)	68.7 (1.0)	69.0 (1.4)	67.1 (1.3)	92.4 (0.2)	92.3 (0.1)
4uni	79.0 (1.1) ▲	66.5 (2.1) ▲	65.7 (1.8)	47.0 (3.3)	67.8 (1.1)	44.4 (1.7)	75.6 (2.3)	58.5 (2.3)
reut90	65.6 (1.0) ▲	27.7 (2.6) ▲	61.1 (0.5)	14.0 (1.0)	58.6 (0.9)	14.5 (0.9)	61.7 (1.1)	18.9 (2.9)
agnews	90.6 (0.1) ▲	89.9 (0.1) ▲	84.0 (0.2)	82.4 (0.2)	87.1 (0.2)	86.1 (0.2)	73.0 (0.3)	73.1 (0.2)

Table 5.6: Average $MicroF_1$ and $MacroF_1$ (and their 95% confidence interval) of different document representations (P2V, FisherP, PTE) on the RF classifier for Topic Datasets. Statistical tests include BoW and Centroid methods.

Algorithm	Winning Counts		Total
	MicroF_1	MacroF_1	
Hyperwords	4	4	8
PTE	1	1	2
FisherP	0	0	0
BoW	0	0	0
Centroid	0	0	0
Paragraph2Vec	0	0	0

Table 5.7: Number of times each algorithm was a top performer in the evaluation, using the RF classifier, for a total of 5 topic datasets.

almost all datasets. The proposed approach consistently improves the BoW representation, obtaining statistically significant gains over BoW on all datasets. In fact, our Hyperwords representation obtained substantial gains in acm, 4uni and reut90, up to 8% and 18% in MicroF_1 and MacroF_1, respectively, when compared to Bag-of-Words.

One interesting aspect to note about the PTE method is its performance variability depending on the dataset. For 20ng, it outperforms all of the other methods, while for AGnews it is far worse, even when compared to BoW. One advantage of the Bag-of-Hyperwords method is that it consistently performs well, considering both MacroF_1 and MicroF_1, for all collections, weighing in favor of its reliability as a document representation method.

FisherP is the second best baseline. Even with its superior results when compared to Paragraph2vec and Centroid, it is statistically worse than BoW in 20ng, acm and AGnews, with losses up to 11%. FisherP is also inferior to the Hyperwords representation approach on those datasets, with significant MicroF_1 losses of 8% and 11% on

acm and 20ng. Despite its low results in comparison to the proposed approach in most datasets, FisherP results are statistically tied with Hyperwords in reut90 and 4uni. Since 4uni and reut90 are our smallest topic datasets (in terms of the number of documents), we conjecture that the simple and relatively low-dimensional space provided by FisherP is beneficial to those small datasets due to the fact that the small number of features may provide benefits against overfitting.

As expected, FisherP is never worse than Centroid. FisherP is capable of characterizing each document according to its deviation from a Gaussian model, which provides regularization over the space of word embeddings. The empirical benefits of using FisherP over Centroids are substantial on 4uni, acm and 20ng, with gains ranging from 4% to 18%.

Paragraph2vec obtains the worst results. Although appealing, the Paragraph2vec idea did not provide competitive results as previously reported [Le and Mikolov, 2014]. We notice that document representations generated by Paragraph2vec are very sensitive to the order in which documents are presented to the model. In fact, a batch of test documents sorted according to their classes biases the model to good prediction results[9], which may be the case in Le and Mikolov [2014]. Since our datasets are shuffled, present results evaluate Paragraph2vec in more realistic situations.

In order to evaluate the document representation strategies on a different classification paradigm, we also used the Random Forest (RF) as one of the classifiers, since it has been considered by many [Fernández-Delgado et al., 2014; Hastie et al., 2001] as a top-notch supervised algorithm. Tables 5.5 and 5.6 show the RF results on all document representation strategies.

The RF results on Paragraph2vec, FisherP, and Centroid are significantly worse than the obtained with the SVM classifier. This fact provides evidence for the shortcomings of exploiting these representations using the association rules provided by random forest. More specifically, these approaches provide a document representation exploiting direct evidence from the dimensions of each word embedding. We conjecture that those dimensions are noisy and difficult to exploit with algorithms based on recursive partitioning of the space, such as the RF approach.

Considering RF as classifier, the Hyperwords representation is also consistently among the best in 4 out of 5 topic datasets (the only loss happening on the 20ng dataset). Most results are statistically superior to those obtained using BoW, with substantial MacroF_1 gains of about 15% on both 4uni and reut90. As mentioned

[9]We verified that by sorting test documents according to their labels, which produced slightly higher effectiveness results on Paragraph2vec.

before, the Paragraph2vec, FisherP, and Centroid approaches are substantially worse, and the gains of Bag-of-Hyperwords when compared to them achieves up to 97%.

Datasets	Hyperwords		BoW		Centroid	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
aisopos_tw	89.6 (3.2) •	89.3 (3.1) •	89.6 (2.8)	89.3 (2.9)	78.8 (12.5)	79.5 (11.7)
debate	80.1 (1.7) ▲	78.6 (1.8) ▲	77.4 (1.3)	75.0 (2.2)	75.5 (0.6)	62.4 (2.5)
narr_tw	84.6 (2.0) ▲	83.7 (2.2) ▲	81.4 (2.7)	80.4 (2.8)	70.0 (4.2)	64.6 (4.1)
pappas_ted	78.7 (4.1) •	78.1 (4.2) •	76.1 (5.8)	75.5 (5.7)	73.7 (6.5)	70.7 (7.6)
pang_movie	77.0 (1.2) •	77.0 (1.2) •	76.9 (1.3)	76.9 (1.3)	66.0 (0.8)	58.1 (4.0)
sanders_tw	84.7 (2.5) •	84.6 (2.4) •	84.5 (3.1)	84.4 (3.1)	68.5 (2.7)	67.3 (2.9)
ss_bbc	86.7 (4.5) •	58.8 (5.2) •	87.4 (5.6)	54.2 (5.9)	87.2 (6.0)	62.4 (2.3)
ss_digg	80.7 (1.4) •	73.4 (3.8) ▲	77.6 (3.7)	65.3 (3.6)	81.7 (5.1)	60.7 (2.7)
ss_myspace	87.4 (1.4) •	71.8 (5.1) ▲	86.2 (2.8)	66.9 (4.9)	84.4 (1.7)	60.9 (0.9)
ss_rw	78.0 (4.6) •	72.8 (6.7) •	76.0 (4.1)	70.8 (4.7)	76.6 (3.9)	64.8 (6.5)
ss_twitter	76.2 (1.9) ▲	75.1 (1.7) ▲	72.8 (2.8)	71.5 (2.8)	71.9 (3.2)	66.5 (9.2)
ss_youtube	84.0 (2.6) ▲	81.1 (3.1) ▲	81.2 (2.5)	77.4 (3.4)	72.4 (1.3)	57.8 (0.8)
stanford_tw	85.8 (4.2) •	85.7 (4.2) •	84.7 (3.0)	84.5 (3.0)	80.7 (8.2)	80.5 (8.1)
semeval_tw	82.7 (1.9) •	76.8 (3.2) ▼	80.2 (1.6)	71.4 (2.8)	84.6 (0.9)	81.0 (1.6)
vader_amzn	74.4 (1.0) •	73.2 (0.9) •	74.1 (1.4)	72.6 (1.5)	68.8 (1.7)	60.9 (2.1)
vader_movie	79.1 (1.2) •	79.1 (1.2) •	78.2 (0.6)	78.2 (0.6)	67.0 (1.4)	60.5 (7.6)
vader_nyt	68.8 (1.6) •	68.3 (1.5) ▲	67.1 (1.2)	66.2 (1.1)	70.3 (1.3)	61.9 (4.4)
vader_tw	86.8 (1.2) ▲	83.9 (1.7) ▲	84.0 (1.1)	80.4 (1.4)	75.0 (1.0)	59.7 (0.6)
yelp_review	95.2 (0.4) ▲	95.2 (0.4) ▲	93.3 (0.4)	93.3 (0.4)	67.4 (1.7)	62.2 (7.2)

Table 5.8: Average $MicroF_1$ and $MacroF_1$ of different document representations (BoW, Centroid) on the SVM classifier for Sentiment Datasets.

Datasets	Hyperwords		Paragraph2vec		FisherP		PTE	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
aisopos_tw	89.6 (3.2) •	89.3 (3.1) •	56.8 (9.4)	53.7 (11.4)	80.3 (6.7)	79.8 (6.5)	63.5 (5.1)	61.1 (4.1)
debate	80.1 (1.7) •	78.6 (1.8) ▲	67.1 (1.5)	60.7 (2.9)	79.0 (2.5)	76.3 (2.8)	73.6 (4.4)	70.9 (4.5)
narr_tw	84.6 (2.0) •	83.7 (2.2) •	64.7 (4.4)	59.3 (5.6)	82.1 (2.4)	81.0 (2.6)	75.3 (2.2)	73.6 (2.4)
pappas_ted	78.7 (4.1) •	78.1 (4.2) •	61.4 (5.6)	59.9 (6.2)	77.2 (5.5)	76.6 (5.3)	79.0 (2.6)	78.3 (2.6)
pang_movie	77.0 (1.2) •	77.0 (1.2) •	62.4 (1.3)	62.3 (1.5)	77.5 (1.1)	77.5 (1.1)	72.8 (0.9)	72.8 (0.9)
sanders_tw	84.7 (2.5) •	84.6 (2.4) •	71.5 (1.6)	70.7 (2.2)	81.3 (1.8)	81.0 (1.9)	75.2 (2.5)	75.1 (2.5)
ss_bbc	86.7 (4.5) •	58.8 (5.2) •	86.9 (6.5)	46.5 (1.8)	87.8 (5.3)	52.2 (6.1)	86.7 (3.8)	59.1 (10.1)
ss_digg	80.7 (1.4) •	73.4 (3.8) ▲	73.0 (2.1)	42.2 (0.7)	77.9 (2.3)	63.2 (5.7)	74.7 (3.9)	58.8 (4.9)
ss_myspace	87.4 (1.4) •	71.8 (5.1) •	83.5 (3.2)	48.7 (4.0)	87.2 (2.2)	65.1 (3.0)	84.3 (1.2)	61.3 (6.3)
ss_rw	78.0 (4.6) •	72.8 (6.7) •	68.2 (3.2)	42.7 (2.2)	80.7 (4.5)	74.6 (6.2)	68.2 (3.5)	54.8 (6.1)
ss_twitter	76.2 (1.9) •	75.1 (1.7) •	61.2 (3.1)	56.6 (3.5)	75.8 (3.6)	74.6 (3.8)	66.9 (3.1)	65.1 (2.7)
ss_youtube	84.0 (2.6) ▲	81.1 (3.1) ▲	73.2 (2.2)	63.8 (2.9)	81.8 (2.6)	77.5 (3.4)	76.8 (1.2)	70.2 (1.3)
stanford_tw	85.8 (4.2) •	85.7 (4.2) •	50.4 (9.3)	49.2 (8.2)	79.1 (2.8)	78.8 (2.9)	72.2 (6.4)	72.1 (6.6)
semeval_tw	82.7 (1.9) •	76.8 (3.2) ▼	71.8 (2.7)	47.4 (1.7)	81.3 (1.4)	73.5 (2.4)	74.1 (1.9)	63.0 (3.9)
vader_amzn	74.4 (1.0) •	73.2 (0.9) •	64.8 (3.0)	62.3 (3.2)	74.0 (1.4)	72.8 (1.4)	69.9 (0.9)	68.0 (0.7)
vader_movie	79.1 (1.2) •	79.1 (1.2) •	61.9 (1.1)	61.9 (1.1)	78.5 (0.9)	78.5 (0.9)	72.4 (1.2)	72.4 (1.2)
vader_nyt	68.8 (1.6) •	68.3 (1.5) •	53.6 (4.9)	46.9 (6.3)	70.3 (2.4)	69.7 (2.5)	61.1 (2.1)	60.3 (2.0)
vader_tw	86.8 (1.2) ▲	83.9 (1.7) ▲	72.3 (1.3)	60.1 (1.8)	85.6 (1.3)	81.9 (1.3)	80.0 (1.1)	75.0 (1.7)
yelp_review	95.2 (0.4) ▲	95.2 (0.4) ▲	90.9 (0.6)	90.9 (0.6)	93.4 (0.6)	93.4 (0.6)	92.0 (0.7)	92.0 (0.7)

Table 5.9: Average $MicroF_1$ and $MacroF_1$ of different document representations (P2V, FisherP, PTE) on the SVM classifier for Sentiment Datasets. Statistical tests include BoW and Centroid methods.

5.2.2 Sentiment Datasets

We evaluate the document representation approaches on a wide range of sentiment datasets. The classification challenge in this domain lies on the fact that documents in most such datasets contain only a small number of words (i.e. low density on Table 5.1)

Algorithm	Winning Counts		Total
	MicroF_1	MacroF_1	
Hyperwords	19	18	37
FisherP	11	8	19
BoW	8	6	14
Centroid	4	2	6
PTE	2	2	4
Paragraph2Vec	1	0	1

Table 5.10: Number of times each algorithm was a top performer in the evaluation, using the SVM classifier, for a total of 19 sentiment datasets.

and the datasets are much smaller, which gives less information for the learning process of the classifiers.

Considering all approaches, Table 5.8 and 5.9 show that Bag-of-Hyperwords is the only document representation capable of consistently achieving the best results in all nineteen datasets (tying (•) or being statistically superior (▲) to the best among the baselines).

Interestingly, the PTE method, whose performance was competitive for the topic datasets, did not perform well for the sentiment datasets. We conjecture that this is due to the information contained in these small datasets being not enough for the PTE to generate good co-occurrence statistics (thus yielding overfitted word vectors), resulting in poor classification performance.

The good $MacroF_1$ results obtained by the Bag-of-Hyperwords model indicates that the additional evidence provided by the word embeddings is capable of enriching the representation of documents in the smaller classes with additional discriminative information. This is particularly interesting, since the extra information obtained by using Hyperwords might be complementing the rare but potentially discriminative words that appear in documents of the smaller classes.

5.2.3 Combination of Hyperwords and Fisher Pooling

We also analyze the combination of our approach and a pooling strategy, since they exploit information from word embeddings in completely different ways. More specifically, while Fisher pooling exploits direct evidence from the vector representation of words, the Bag-of-Hyperwords representation uses word embeddings only as a source of similarity between words. Due to their differences, both representations might provide complementary discriminative evidence for classification. For this experiment, we aggregate the features from the Fisher pooling method with the hyperword features (i.e. a concatenation of both feature vectors) to generate a new representation for the documents.

Datasets	Hyperwords		Hyperwords+FisherP	
	$MicF_1$	$MacF_1$	$MicF_1$	$MacF_1$
acm	74.0 (0.8)	61.9 (1.8)	74.6 (1.5) ●	62.1 (7.5) ●
20ng	91.1 (0.9)	91.0 (0.9)	91.1 (1.2) ●	91.0 (1.2) ●
4uni	75.6 (3.4)	64.0 (2.6)	81.0 (2.1) ▲	68.5 (3.5) ▲
reut90	68.9 (1.5)	30.8 (3.6)	67.5 (1.8) ●	30.2 (4.6) ●
agnews	91.5 (0.1)	90.9 (0.1)	91.6 (0.1) ▲	91.0 (0.1) ▲
aisopos_tw	89.6 (3.2)	89.3 (3.1)	89.6 (3.2) ●	89.3 (3.2) ●
debate	80.1 (1.7)	78.6 (1.8)	78.9 (3.4) ●	76.9 (3.9) ●
narr_tw	84.6 (2.0)	83.7 (2.2)	84.4 (1.4) ●	83.4 (1.7) ●
pappas_ted	78.7 (4.1)	78.1 (4.2)	77.9 (5.7) ●	77.4 (5.6) ●
pang_movie	77.0 (1.2)	77.0 (1.2)	77.8 (0.8) ▲	77.8 (0.8) ▲
sanders_tw	84.7 (2.5)	84.6 (2.4)	84.2 (2.4) ●	84.0 (2.4) ●
ss_bbc	86.7 (4.5)	58.8 (5.2)	87.3 (4.8) ●	55.9 (7.6) ●
ss_digg	80.7 (1.4)	73.4 (3.8)	78.7 (3.8) ●	69.3 (5.9) ●
ss_myspace	87.4 (1.4)	71.8 (5.1)	87.0 (3.0) ●	68.3 (5.9) ●
ss_rw	78.0 (4.6)	72.8 (6.7)	80.7 (2.3) ●	76.0 (3.5) ●
ss_twitter	76.2 (1.9)	75.1 (1.7)	76.3 (2.2) ●	75.0 (2.5) ●
ss_youtube	84.0 (2.6)	81.1 (3.1)	84.8 (1.5) ●	81.6 (1.8) ▲
stanford_tw	85.8 (4.2)	85.7 (4.2)	79.4 (3.9) ▼	79.1 (4.0) ▼
semeval_tw	82.7 (1.9)	76.8 (3.2)	81.7 (2.0) ●	74.9 (3.4) ●
vader_amzn	74.4 (1.0)	73.2 (0.9)	74.6 (1.1) ▲	73.5 (1.2) ▲
vader_movie	79.1 (1.2)	79.1 (1.2)	79.8 (0.7) ▲	79.7 (0.7) ▲
vader_nyt	68.8 (1.6)	68.3 (1.5)	69.4 (2.5) ●	69.0 (2.4) ●
vader_tw	86.8 (1.2)	83.9 (1.7)	87.7 (0.8) ▲	84.7 (0.5) ▲
yelp_review	95.2 (0.4)	95.2 (0.4)	94.7 (0.3) ●	94.7 (0.3) ●

Table 5.11: Average $MicroF_1$ and $MacroF_1$ (and their 95% confidence interval) of the two final proposals on the SVM classifier.

Algorithm	Winning Counts		Total
	MicroF_1	MacroF_1	
Hyperwords+FisherP	23	23	46
Hyperwords	18	17	35

Table 5.12: Number of times each algorithm was a top performer in the evaluation, using the SVM classifier, for a total of 24 topic and sentiment datasets.

Table 5.11 presents the results of the Hyperwords representation method, as well as the combination Hyperwords+FisherP. Considering topic categorization and sentiment analysis datasets, Hyperwords+FisherP achieves results that are superior to or statistically tied with Hyperwords in most cases, which provides evidence towards our hypothesis of complementary information between the two representations. The only exception is the sentiment analysis dataset stanford_tw, which is one of the smallest (and therefore susceptible to overfitting) datasets.

5.3 Effects of the Parameters

So far, we have compared the Bag-of-Hyperwords model with other document representation methods that also rely on word embeddings. We also have analyzed how to the use of Hyperwords (specially those that carry semantic information) can lead to

improvements on the discriminative power of both individual features and overall representation, when compared to the traditional Bag-of-Words model. In this section, we study the effects of the parameters of the Bag-of-Hyperwords, namely, α, that controls the degree of specificity of the concepts captured into Hyperwords; β, that controls how aggressively similar Hyperwords are merged together; and, finally, D (Dynamic α Selection), which indicates whether all Hyperwords follow the same global α (following Eq. 4.2) or if each Hyperword has its own optimal α (following Eq. 4.8). In order to evaluate the effects of the parameters, we perform a $2^k r$-factorial design with replication.

5.3.1 Background: The 2^k and $2^k r$-factorial designs

According to Jain [1991], 2^k-factorial experimental designs are used to determine the effect of k factors, each of which have two alternative levels. He argues that very often, the effect of a factor is unidirectional, that is, the performance either continually increases as the factor increases from minimum to maximum, and vice-versa. He uses the performance of a computer system as an example: the performance is expected to increase as the memory size, or the computing power increases. We describe the process of computing the effects for two factors below:

Consider two factors, A and B, for which one wants to analyze the effects over some response variable that depends on those two factors. It is necessary to perform $2^2 = 4$ evaluations of the response variable, as follows:

y_1: Response for $\mathrm{A_{low}}$, $\mathrm{B_{low}}$
y_2: Response for $\mathrm{A_{high}}$, $\mathrm{B_{low}}$
y_3: Response for $\mathrm{A_{low}}$, $\mathrm{B_{high}}$
y_4: Response for $\mathrm{A_{high}}$, $\mathrm{B_{high}}$

For each factor, as well as for the interactions between them, a ***contrast*** is created. From the observations above, the contrasts are computed as follows:

$$\begin{aligned} q_0 &= \frac{1}{2^2}(y_1 + y_2 + y_3 + y_4) \\ q_{\mathrm{A}} &= \frac{1}{2^2}(-y_1 + y_2 - y_3 + y_4) \\ q_{\mathrm{B}} &= \frac{1}{2^2}(-y_1 - y_2 + y_3 + y_4) \\ q_{\mathrm{AB}} &= \frac{1}{2^2}(y_1 - y_2 - y_3 + y_4) \end{aligned}$$

Which encodes a linear model that can be expressed by the following equation:

$$\hat{y} = q_0 + q_A x_A + q_B x_B + q_{AB} x_{AB} \tag{5.5}$$

for which $x_A = -1$ when using A_{low}, $x_A = 1$ when using A_{high}. The same logic applies to B, and $x_{AB} = x_A \cdot x_B$.

Interesting conclusions can be drawn from observing the contrasts. For example, a positive contrast indicates that the factor contributes positively to the response variable, while a negative contrast indicates the opposite. More importantly, the contrasts can be used to compute the allocation of variation for each factor (and their interactions). This can be achieved as follows:

$$\text{Fraction of variation explained by A} := \frac{\text{SSA}}{\text{SST}}$$

$$\begin{aligned} \text{SSA} &= 2^2 q_A^2 \\ \text{SSB} &= 2^2 q_B^2 \\ \text{SSAB} &= 2^2 q_{AB}^2 \\ \text{SST} &= \text{SSA} + \text{SSB} + \text{SSAB} \end{aligned} \tag{5.6}$$

5.3.1.1 The $2^k r$-factorial design

The $2^k r$-factorial design shares many similarities with the 2^k-factorial design. The difference, in principle, is that for every combination of factor levels, multiple observations of the response variable are observed. This gives space to variability, as some of the variance cannot be explained only by the factors under study. Therefore, it can be defined by introducing the error factor into equations 5.5 and 5.6:

$$\hat{y} = q_0 + q_A x_A + q_B x_B + q_{AB} x_{AB} + e \tag{5.7}$$

$$\begin{aligned} \text{SSA} &= 2^2 r q_A^2 \\ \text{SSB} &= 2^2 r q_B^2 \\ \text{SSAB} &= 2^2 r q_{AB}^2 \\ \text{SSE} &= \sum_{i=1}^{2^2} \sum_{j=1}^{r} (y_{ij} - \bar{y}_i)^2 \\ \text{SST} &= \text{SSA} + \text{SSB} + \text{SSAB} + \text{SSE} \end{aligned} \tag{5.8}$$

5.3.2 Evaluating α, β and Dynamic Alpha Selection

We perform the $2^k r$-factorial design to evaluate the effects of α, β and the Dynamic Alpha Selection mechanism on the performance of Bag-of-Hyperwords. We choose the following levels for the factors: $\alpha = \{0.4, 0.8\}$ and $\beta = \{0.3, 0.7\}$. Dynamic Alpha Selection (which we will represent simply as D) is a binary option. Therefore, D={disabled, enabled}. We evaluate the effects for 2 topic datasets and 2 sentiment datasets. Namely, acm, 4uni, pang_movie and vader_tw. For compactness, we omit the computation of the effects and present only their final value (as well as their signs) in table 5.13.

Factor:	**Dataset:**	acm	webkb	pang_movie	vader_tw
α	Fr. of Var. (%)	6.0	17.9	50.0	53.1
	sign	−	−	−	−
β	Fr. of Var. (%)	5.2	22.1	5.1	1.9
	sign	−	−	−	−
D	Fr. of Var. (%)	29.5	0.0	0.0	3.4
	sign	+	−	−	−
$\alpha\beta$	Fr. of Var. (%)	4.4	20.6	5.7	1.6
	sign	+	+	+	+
αD	Fr. of Var. (%)	2.8	24.0	14.6	18.7
	sign	+	−	−	−
βD	Fr. of Var. (%)	13.6	1.8	2.3	2.4
	sign	+	+	+	+
$\alpha\beta D$	Fr. of Var. (%)	14.1	2.3	1.3	1.6
	sign	−	−	−	−
Error	Fr. of Var. (%)	24.4	11.3	21.9	17.2

Table 5.13: Allocation of variance for α, β and Dynamic Alpha Selection(D). The allocation is expressed as a percentual fraction.

Table 5.13 shows that for sentiment datasets (pang_movie and vader_tw), the most important factor is α. Also, for all datasets, the lower the value of α, the better is the performance of the Bag-of-Hyperwords model (evaluated using MacroF_1 of a SVM classifier). Recall that lower values of α correspond to bigger Hyperwords, so as α decreases, more distant the representation becomes from the Bag-of-Words model. This supports the claim that Bag-of-Hyperwords enriches the document representation for almost every dataset, when combined with the Bag-of-Words.

Another interesting observation is that the effect of the interaction between the parameters corresponds to a large fraction of the allocation of variance. This is not a desirable discovery, because defining ideal parameters is easier when the effect of the interaction between parameters is low. But as a rule of thumb, good results are obtained

when α is as low as possible, β is also as low as possible and the Dynamic Alpha Selection is enabled (which can be verified by examining the tables in Appendix A), although a full search of every combination of the parameters is recommended when the optimal results are needed.

5.4 Explaining the Improvement over Bag-of-Words

In this section we conduct experiments in order to explain why the Bag-of-Hyperwords model consistently outperforms the traditional Bag-of-Words model. For the following experiments, we work under the hypothesis that the overall improvement of the model as a whole can be partially explained by the individual improvement of each of the models' features. For simplicity, the original hyperwords (i.e., before merging, without dynamic alpha selection) are used for the experiments conducted in this section. Since a single dataset is capable of providing a sufficient number of features to make statistically significant comparisons, we use the ACM dataset for the experiments that follow. We argue that any dataset for which the improvement in $\text{MicroF}_1/\text{MacroF}_1$ is significant can be used to draw the same conclusions.

5.4.1 Feature Space Mapping

In order to understand the experiments that follow, we define what we mean by individual feature improvement. In order to do so, we must draw a 1:1 correspondence between features in Bag-of-Words space and Bag-of-Hyperwords space (notice that we are using the set of hyperwords obtained before merging). We make this correspondence in a straightforward manner: recall that for every term t in the BoW feature space, we generate a hyperword h_t, according to equation Eq 4.2. Then, for each of the hyperwords, we compute their TF and IDF weights. Therefore, for each t in Bag-of-Words space, there is h_t in Bag-of-Hyperwords space, and vice-versa.

5.4.1.1 Semantics and Implications of $t : h_t$ Mapping

The semantics of the considered $t : h_t$ feature space mapping can be thought of as a relation from each term to a set that represents the union of every concept that is related to t. In other words, the feature corresponding to h_t in Bag-of-Hyperwords space can be seen as the general concept expressed by term t. Consequently, h_t will appear in at least every document t appears, and possibly some more documents.

5.4.2 Evaluating Individual Features: Mutual Information

Evaluating features individually requires that some score is defined for each feature. In the context of Automated Text Classification, we want the score of each feature to reflect how descriptive/informative each feature is with respect to the category of each document. A useful measure called Mutual Information or Information Gain can be applied for this matter. Formally, let each feature correspond to a random variable $X_i \in \mathbf{X}$ and the category attribute correspond to the random variable Y. The Mutual Information between X_i and Y is defined as follows [MacKay, 2002]:

$$\mathrm{MI}(Y; X_i) = H(Y) - H(Y|X_i) \tag{5.9}$$

For our experiments, we consider $\mathbf{X}$ and Y to be discrete random variables. The definitions of their probability functions are given below:

$$p(Y = y) = \frac{1}{|\mathbb{D}|} \sum_{d \in \mathbb{D}} I[\mathrm{category}(d) = y] \tag{5.10}$$

$$p(Y = y|X_i = T) = \frac{1}{|\{d \in \mathbb{D} : X_i \in d\}|} \sum_{d \in \mathbb{D}: X_i \in d} I[\mathrm{category}(d) = y] \tag{5.11}$$

$$p(Y = y|X_i = F) = \frac{1}{|\{d \in \mathbb{D} : X_i \notin d\}|} \sum_{d \in \mathbb{D}: X_i \notin d} I[\mathrm{category}(d) = y] \tag{5.12}$$

Where $\mathbb{D}$ is the collection of documents, $I[\mathrm{predicate}]$ is the Indicator Function (equivalent to 1 if predicate is true, 0 otherwise). $X_i \in d$ is true if, and only if, the term corresponding to X_i occurs in document d. The next set of equations define the entropy and conditional entropy for the aforementioned variables:

$$H(Y) = -\sum_{y \in Y} p(Y = y) \log p(Y = y) \tag{5.13}$$

$$\begin{aligned} H(Y|X_i) = &- p(X_i = T) \sum_{y \in Y} p(Y = y|X_i = T) \log p(Y = y|X_i = T) \\ &- p(X_i = F) \sum_{y \in Y} p(Y = y|X_i = F) \log p(Y = y|X_i = F) \end{aligned} \tag{5.14}$$

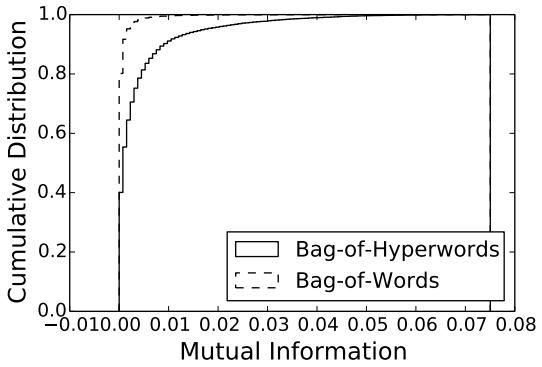
5.4.3 Difference in Expected Mutual Information

In this section, we provide and analyze empirical evidence supporting that the expected mean mutual information for features in Bag-of-Hyperwords space is significantly greater than that obtained using features in the simple Bag-of-Words value,

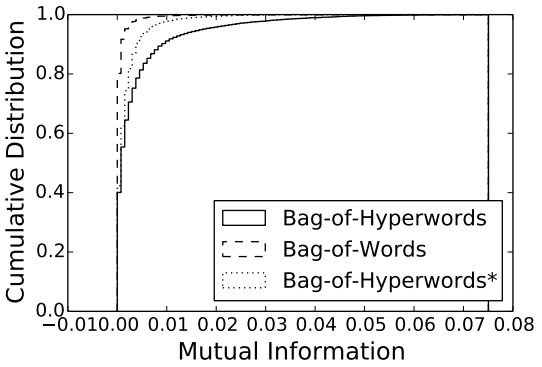
with confidence over 99%. Table 5.14 contains the confidence intervals for the expected Mutual Information of features in each space, with p-values for the Wilcoxon signed-rank test and the Student's t-test equal to 0 (with float precision) under the null hypothesis that the Mutual Information values come from the same population.

Feature Space	Mean Mutual Information
Bag-of-Words	$(6.62 \times 10^{-4}, 7.20 \times 10^{-4})$
Bag-of-Hyperwords	$(37.22 \times 10^{-4}, 40.02 \times 10^{-4})$

Table 5.14: Mean Mutual Information of features in Bag-of-Words and Bag-of-Hyperwords spaces and their 99% Confidence Intervals.



(a) Cumulative Distribution Functions for the Mutual Information of Bag-of-Words features and Bag-of-Hyperwords features.



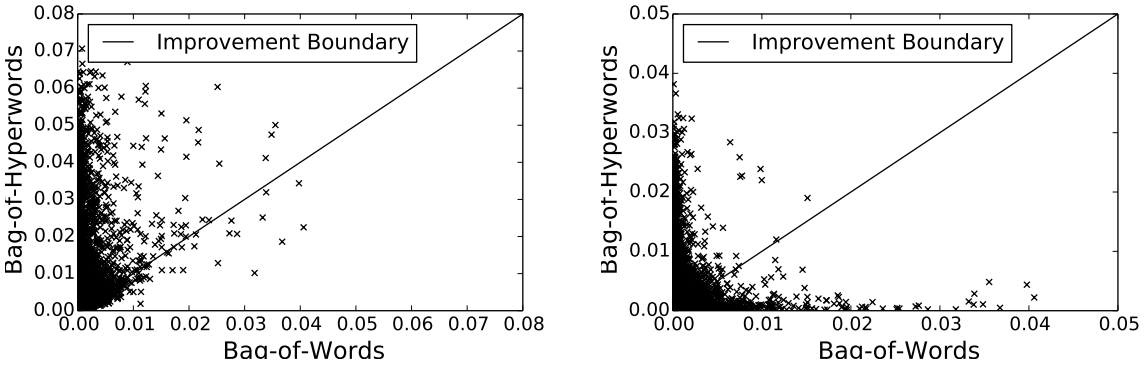
(b) Cumulative Distribution Functions for the Mutual Information of Bag-of-Words features and Bag-of-Hyperwords features. Here, the distinction between Hyperwords obtained from Word Embeddings cosine similarities vs. Semantic-less Hyperwords (marked with *. See section 5.4.4.1 for details).

Figure 5.1: Cumulative Distribution Functions for the Mutual Information of features from Bag-of-Words vs. Bag-of-Hyperwords

Not only the Mean Mutual Information is expected to be higher, we observe that the Cumulative Distribution Function for Hyperwords dominates the CDF for Bag-of-Words features. This means that, no matter what percentile of features (from smallest MI to greatest MI), the maximum MI found for Hyperwords is greater than that found for Bag-of-Words features. This is shown in figure 5.1a.

5.4.4 The Role of Semantic Similarity in Hyperwords

The experiments shown in section 5.4.3 are sufficient to demonstrate the superiority of the features obtained from the Bag-of-Hyperwords model in terms of Mutual Infor-



(a) Scatter plot of Mutual Information for each feature in Bag-of-Words versus in Bag-of-Hyperwords space.

(b) Scatter plot of Mutual Information for each feature in Bag-of-Words versus in Bag-of-Hyperwords space, for Semantic-less Hyperwords (see section 5.4.4.1).

Figure 5.2: Scatter plots for Mutual Information of individual features for Bag-of-Words vs. Bag-of-Hyperwords.

mation with respect to the category attribute for Automated Document Classification. We now want to demonstrate that the semantic information contained in Hyperwords plays a fundamental role in the Mutual Information increase. In order to demonstrate the validity of this hypothesis, we introduce Hyperwords that do not necessarily carry semantic similarity information (henceforth called Semantic-less Hyperwords) and compare them with Hyperwords obtained from Word2Vec cosine similarities.

5.4.4.1 Generating Semantic-less Hyperwords

Generating semantic-less Hyperwords (i.e., Hyperwords that do not carry semantic similarity information) is a tricky task. This is due to the fact that many variables need to be controlled for the artificial hyperwords be comparable to the Hyperwords generated from real world similarities between word pairs. For instance, it is desirable that the word similarity values follow the same statistic distribution, and the same holds for hyperword sizes and other characteristics inherent to the Hyperwords model.

We propose a technique to overcome the aforementioned difficulties while still removing semantic similarity information from hyperwords. It is a simple approach that achieves the desired results, described as follows:

---

1. Let $F(t) \rightarrow t'$ be a function that maps each $t \in \mathbb{V}$ to a fixed, randomly chosen term $t' \in \mathbb{V}$.

2. From a set of existing Hyperwords $\{h_t \in \mathcal{H}\}$:

a) For each $h_t = (w_1, w_2, ..., w_{|\mathbb{V}|})$, create $h'_t = (w_{F(1)}, w_{F(2)}, ..., w_{F(|\mathbb{V}|)})$

b) Create $\mathcal{H}' = \{h'_t\}_{1:|\mathbb{V}|}$.

In practice, each word inside a Hyperword has its weight exchanged with some other fixed randomly chosen word. This process removes the semantic relationship from words because otherwise it would mean that semantic relationships can be constructed by grouping random words together. Furthermore, the only meaningful distinction between $\mathcal{H}'$ and $\mathcal{H}$ is, by construction, the semantics of the hyperwords.

We demonstrate, with over 99% confidence, that Hyperwords that carry semantic meaning have features with significantly higher Mutual Information when compared to Semantic-less Hyperwords (Table 5.15). This is a strong evidence that both (a) Hyperwords generated from Word2Vec similarities carry semantic information; and (b) the presence of semantic information leads to an increase in the expected Mutual Information of each feature. Figure 5.2 shows the how more features are improved (and less features are worsened) for the Bag-of-Hyperwords (Fig. 5.2a) when compared to the Bag-of-(Semantic-less)-Hyperwords (Fig. 5.2b).

Feature Space	**Mean Mutual Information**
Bag-of-Hyperwords	$(3.72 \times 10^{-3}, 4.00 \times 10^{-3})$
Bag-of-(Semantic-less)-Hyperwords	$(1.90 \times 10^{-3}, 2.01 \times 10^{-3})$

Table 5.15: Mean Mutual Information of features in Bag-of-Hyperwords and Bag-of-(Semantic-less)-Hyperwords spaces and their 99% Confidence Intervals.

Chapter 6

Conclusions and Future Work

In this dissertation, we proposed a new way of representing documents to be used in text classification tasks. Our method is able to combine the simplicity and effectiveness of the traditional BoW representation with semantic information derived from word embeddings. This combination is performed by creating new features (hyperwords) resulting from the clustering of semantic correlated terms, that is, terms with high similarities regarding the corresponding Word2Vec representations. We devised methods to assign TF-IDF values to these new features, merging redundant hyperwords together, and automatically selecting the best level of specificity for each hyperword. The resulting representation has a great advantage of being able to be directly incorporated by any classifier that can already handle the BoW representation, and be combined with complementary feature sets, for example that obtained from applying the Fisher Vector method on Word Embeddings.

As our experiments show, the proposed Bag-of-Hyperwords presented consistently better classification results than the traditional BoW representation. Also, our proposed document representation was shown to be statistically equivalent or superior to best methods that exploit Word2Vec information in most cases. We also combined our hyperwords representation with the Fisher pooling representation and obtained results comparable to (and sometimes better than) both methods considered separately. Thus, we can conclude that our proposed representation is a promising document representation strategy for the task of document classification and related ones (e.g., query expansion, tag recommendation). Additionally, we compare the use of the Word2Vec vectors precomputed from the Google News dataset with the vectors obtained from running PTE on the collection. Our experiments suggest that using precomputed vectors is preferable for two main reasons: (1) they can be easily obtained and comprise an extensive vocabulary, facilitating its application to real-world datasets of varying

sizes; and (2) it provides embeddings that are not biased by occurrence counts of each particular collection, resulting in better generalization and reducing the chance of overfitting. We also performed a study demonstrating that hyperwords are more discriminative than simple words. We also demonstrated that our method is capable of extracting semantic information from word embeddings, and that that semantic information plays a fundamental role on the overall quality of the obtained hyperwords.

There are many extensions to our work. Regarding the generation of hyperwords, we have only experimented with a simple clustering approach both when obtaining the initial hyperwords and in the hyperword clustering process. Several other clustering approaches can be used. Also, we experimented with only one similarity measure (i.e., cosine). It would be interesting to investigate other similarity measures such as Euclidean, Manhattan or L2, or even a combination of several similarity features over several word embedding strategies.

Another interesting line of investigation is to evaluate the use of feature selection techniques to be applied in our bag of hyperwords representation, aiming at diminishing the number of features and selecting the best features among the hyperwords. Finally, we only experimented our enhanced document representation with two classifiers (SVM and Random Forest). A more comprehensive study could be achieved by extending our experiments to other state-of-the-art classifiers (e.g., Salles et al. [2015]; Chen and Guestrin [2016]) to evaluate how the generated feature sets influence the effectiveness of different classification methods.

Appendix A

Detailed Computation of Effects of Factors

I	α	β	D	$\alpha\beta$	αD	βD	$\alpha\beta D$	$\bar{y}$
1	−1	−1	−1	1	1	1	−1	0.7448
1	1	−1	−1	−1	−1	1	1	0.7054
1	−1	1	−1	−1	1	−1	1	0.6982
1	1	1	−1	1	−1	−1	−1	0.7050
1	−1	−1	1	1	−1	−1	1	0.7303
1	1	−1	1	−1	1	−1	−1	0.7337
1	−1	1	1	−1	−1	1	−1	0.7424
1	1	1	1	1	1	1	1	0.7328
5.7926	−0.0388	−0.0359	0.0858	0.0333	0.0265	0.0583	−0.0592	*Total*
0.7241	−0.0048	−0.0045	0.0107	0.0042	0.0033	0.0073	−0.0074	*Contrast*

Table A.1: Effects for $2^k r$-factorial design on acm.

I	α	β	D	$\alpha\beta$	αD	βD	$\alpha\beta D$	$\bar{y}$
1	−1	−1	−1	1	1	1	−1	0.7447
1	1	−1	−1	−1	−1	1	1	0.7128
1	−1	1	−1	−1	1	−1	1	0.6730
1	1	1	−1	1	−1	−1	−1	0.7129
1	−1	−1	1	1	−1	−1	1	0.7567
1	1	−1	1	−1	1	−1	−1	0.6846
1	−1	1	1	−1	−1	1	−1	0.7187
1	1	1	1	1	1	1	1	0.6825
5.6860	−0.1004	−0.1116	−0.0009	0.1077	−0.1162	0.0316	−0.0360	*Total*
0.7107	−0.0125	−0.0140	−0.0001	0.0135	−0.0145	0.0039	−0.0045	*Contrast*

Table A.2: Effects for $2^k r$-factorial design on 4uni.

I	α	β	D	$\alpha\beta$	αD	βD	$\alpha\beta D$	$\bar{y}$
1	−1	−1	−1	1	1	1	−1	0.7722
1	1	−1	−1	−1	−1	1	1	0.7493
1	−1	1	−1	−1	1	−1	1	0.7472
1	1	1	−1	1	−1	−1	−1	0.7485
1	−1	−1	1	1	−1	−1	1	0.7757
1	1	−1	1	−1	1	−1	−1	0.7348
1	−1	1	1	−1	−1	1	−1	0.7690
1	1	1	1	1	1	1	1	0.7365
6.0331	−0.0950	−0.0307	−0.0014	0.0325	−0.0519	0.0207	−0.0159	*Total*
0.7541	−0.0119	−0.0038	−0.0002	0.0041	−0.0065	0.0026	−0.0020	*Contrast*

Table A.3: Effects for $2^k r$-factorial design on pang_movie.

I	α	β	D	$\alpha\beta$	αD	βD	$\alpha\beta D$	$\bar{y}$
1	−1	−1	−1	1	1	1	−1	0.8577
1	1	−1	−1	−1	−1	1	1	0.8224
1	−1	1	−1	−1	1	−1	1	0.8227
1	1	1	−1	1	−1	−1	−1	0.8201
1	−1	−1	1	1	−1	−1	1	0.8556
1	1	−1	1	−1	1	−1	−1	0.7812
1	−1	1	1	−1	−1	1	−1	0.8568
1	1	1	1	1	1	1	1	0.7824
6.5989	−0.1866	−0.0350	−0.0470	0.0327	−0.1108	0.0398	−0.0327	*Total*
0.8249	−0.0233	−0.0044	−0.0059	0.0041	−0.0139	0.0050	−0.0041	*Contrast*

Table A.4: Effects for $2^k r$-factorial design on vader_tw.

Bibliography

Amiri, H. and III, H. D. (2016). Short text representation for detecting churn in microblogs. In Schuurmans, D. and Wellman, M. P., editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2566--2572. AAAI Press.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Canuto, S., Gonçalves, M. A., and Benevenuto, F. (2016). Exploiting new sentiment-based meta-level features for effective sentiment analysis. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 53--62, New York, NY, USA. ACM.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785--794, New York, NY, USA. ACM.

Diakopoulos, N. A. and Shamma, D. A. (2010). Characterizing debate performance via aggregated twitter sentiment. In *SIGCHI'10*, pages 1195--1198. ACM.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133--3181. ISSN 1532-4435.

Fotis Aisopos (2014). Manually annotated sentiment analysis twitter dataset ntua. `www.grid.ece.ntua.gr`.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. Technical report, Stanford.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Harris, Z. (1954). Distributional structure. *Word*.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Hutto, C. J. and Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM'14*.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Netw.*, 13(4-5):411--430. ISSN 0893-6080.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley professional computing. Wiley. ISBN 978-0-471-50336-1.

Ken Chatfield, Victor Lempitsky, A. V. and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference*, pages 76.1--76.12. BMVA Press. http://dx.doi.org/10.5244/C.25.76.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Lev, G., Klein, B., and Wolf, L. (2015). *Natural Language Processing and Information Systems: 20th International Conference on Applications of Natural Language to Information Systems, NLDB 2015, Passau, Germany, June 17-19, 2015, Proceedings*, chapter In Defense of Word Embedding for Generic Text Representation, pages 35--50. Springer International Publishing, Cham.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *JMLR.*, 5:361--397.

MacKay, D. J. C. (2002). *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA. ISBN 0521642981.

Mesnil, G., Mikolov, T., Ranzato, M., and Bengio, Y. (2014). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *CoRR*, abs/1412.5335.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Narr, S., Hulfenhaus, M., and Albayrak, S. (2012). Language-independent twitter sentiment analysis. *Knowledge Discovery and Machine Learning (KDML)*, pages 12--14.

Palakodety, S. and Callan, J. (2014). Query transformations for result merging. In Voorhees, E. M. and Ellis, A., editors, *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*, volume Special Publication 500-308. National Institute of Standards and Technology (NIST).

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL'04*, Stroudsburg, PA, USA.

Pappas, N. and Popescu-Belis, A. (2013). Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In *SIGIR'13*, pages 773--776. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825--2830.

Peng, X., Zou, C., Qiao, Y., and Peng, Q. (2014). *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, chapter Action Recognition with Stacked Fisher Vectors, pages 581--595. Springer International Publishing, Cham.

Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. ISSN 1063-6919.

Salles, T., Gonçalves, M., Rodrigues, V., and Rocha, L. (2015). Broof: Exploiting out-of-bag errors, boosting and random forests for effective automated classification. In *Proceedings of SIGIR'15*, pages 353--362. ACM.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513--523. ISSN 0306-4573.

Sanchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*, 105(3):222–245.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Tang, J., Qu, M., and Mei, Q. (2015). Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165--1174. ACM.

Thelwall, M. (2013). Heart and soul: Sentiment strength detection in the social web with sentistrength. `http://sentistrength.wlv.ac.uk/documentation/SentiStrengthChapter.pdf`.

Xing, C., Wang, D., Liu, C., and Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In Mihalcea, R., Chai, J. Y., and Sarkar, A., editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1006--1011. The Association for Computational Linguistics.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Inf. Ret.*, 1:69–90. ISSN 1386-4564.

Yuan, Y., He, L., Peng, L., and Huang, Z. (2014). A new study based on word2vec and cluster for document categorization. *Journal of Computational Information Systems*, 10(21):9301--9308.

Zhang, D., Yuan, B., Wang, D., and Liu, R. (2015). Joint semantic relevance learning with text data and graph knowledge. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality CVSC*, pages 32--40.

Index