Novel Representations, Regularization and Distances for Text Classification

Konstantinos Skianis

université PARIS-SACLAY

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

NNT : 2019SACLX012

Thèse de doctorat

Novel Representations, Regularization & Distances for Text Classification

Thèse de doctorat de l'Université Paris-Saclay
préparée à Ecole Polytechnique

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 01/03/2019, par

KONSTANTINOS SKIANIS

Composition du Jury :

Michalis Vazirgiannis Professor, École Polytechnique, France	Directeur de thèse
Guillaume Wisniewski Professor, Universite Paris-Sud, France	Co-directeur de thèse
Ion Androutsopoulos Professor, Athens University of Economics and Business, Greece	Rapporteur
Sameer Singh Professor, University of California, Irvine, USA	Rapporteur
Themis Palpanas Professor, Paris Descartes University, France	Examinateur
Giannis Manolopoulos Professor, Aristotle University, Thessaloniki, Greece	Examinateur
Maks Ovsjanikov Professor, École Polytechnique, France	Président du jury

ABSTRACT

Text is undoubtedly the dominant method for storing, accessing and transferring knowledge since the start of human civilization. With the rise of the computer age, along with the invention of the Internet, text has been established as one of the most popular ways to access and distribute data between humans, sometimes preferred over speech (e.g. text messages over phone calls). Search engines, social networks and online purchase platforms are only some examples of the largest text-using technologies across the globe, creating a huge demand for fast and meaningful use of these data. The need for tools that process text effectively and efficiently is evident.

Modelling text for machines to understand is hard and thus a great research effort is done by the communities of Computational Linguistics (CL), Natural Language Processing (NLP) and Text Mining (TM). The complexity to model language for machines is thus high, as we humans have not even discovered how our brain models text and speech. Trying to mimic a more human-intelligent way, researchers exploited other techniques from areas like Machine Learning (ML) and more generally Artificial Intelligence (AI), in order to tackle the two challenging subparts of NLP, Natural Language Understanding (NLU) and Natural Language Generation (NLG).

Living in the big data era, textual data are also increasing at a high rate. Humans can no longer handle simple tasks, for example even the process of classifying documents, due to the large volume.

Extracting meaningful representations out of text has been a key element for modelling language in order to tackle NLP tasks like text classification. These representations can then form groups that one can use for supervised learning problems. More specifically, one can utilize these linguistic groups for regularization purposes. Last, these structures can be of help in another important field, distance computation between text documents.

The main goal of this thesis is to study the aforementioned problems; first, by examining new graph-based representations of text. Next, we studied how groups of these representations can help regularization in machine learning models for text classification. Last, we dealt with sets and measuring distances between

documents, utilizing our proposed linguistic groups, as well as graph-based approaches.

In the first part of the thesis, we have studied graph-based representations of text. Turning text to graphs is not trivial and has been around even before word embeddings were introduced to the NLP community. In our work, we show that graph-based representations of text can capture effectively relationships like order, semantic or syntactic structure. Moreover, they can be created fast while offering great versatility for multiple tasks.

In the second part, we focused on structured regularization for text. Textual data suffer from the dimensionality problem, creating huge feature spaces. Regularization is critical for any machine learning model, as it can address overfitting. In our work we present novel approaches for text regularization, by introducing new groups of linguistic structures and designing new algorithms.

In the last part of the thesis, we study new methods to measure distance in the word embedding space. First, we introduce diverse methods to boost comparison between documents that consist of word vectors. Next, representing the comparison of the documents as a weighted bipartite matching, we show how we can learn hidden representations and improve results for the text classification task.

Finally, we conclude by summarizing the main points of the total contribution and discuss future directions.

RÉSUMÉ

Le texte est sans aucun doute la méthode dominante de stockage, d'accès et de transfert de connaissances depuis le début de la civilisation humaine. Avec l'avènement de l'ère informatique, ainsi que l'invention d'Internet, le texte a été établi comme l'un des moyens les plus populaires d'accéder et de distribuer des données entre humains, parfois préféré à la parole (par exemple, les SMS sur les appels téléphoniques). Les moteurs de recherche, les réseaux sociaux et les plateformes d'achat en ligne ne sont que quelques exemples des plus grandes technologies d'utilisation de texte à travers le monde, créant une énorme demande pour une utilisation rapide et significative de ces données. Le besoin d'outils pour traiter le texte de manière efficace et efficiente est évident.

Il est difficile de modéliser du texte pour que les machines le comprennent et, par conséquent, un grand effort de recherche est effectué par les communautés de linguistique computationnelle (CL), de traitement du langage naturel (NLP) et d'exploration de texte (TM). La complexité pour modéliser le langage des machines est donc élevée, car nous, les humains, n'avons même pas découvert comment notre cerveau modélise le texte et la parole. En essayant d'imiter une manière plus intelligente pour l'homme, les chercheurs ont exploité d'autres techniques dans des domaines tels que l'apprentissage automatique (ML) et plus généralement l'intelligence artificielle (IA), afin de s'attaquer aux deux sous-parties difficiles de la NLP, la compréhension du langage naturel (NLU) et la nature Génération de langue (NLG).

Vivant à l'ère des mégadonnées, les données textuelles augmentent également à un rythme élevé. Les humains ne peuvent plus gérer des tâches simples, par exemple même le processus de classification des documents, en raison du volume important.

L'extraction de représentations significatives hors du texte a été un élément clé de la modélisation de langage afin de traiter des tâches de la NLP telles que la classification de texte. Ces représentations peuvent ensuite former des groupes que l'on peut utiliser pour des problèmes d'apprentissage supervisé. Plus spécifiquement, on peut utiliser ces groupes linguistiques à des fins de régularisation. Enfin,

ces structures peuvent être utiles dans un autre domaine important, le calcul de distance entre documents texte.

L'objectif principal de cette thèse est d'étudier les problèmes susmentionnés; Tout d'abord, en examinant de nouvelles représentations de texte basées sur des graphes. Ensuite, nous avons étudié comment des groupes de ces représentations peuvent aider à la régularisation dans des modèles d'apprentissage automatique pour la classification de texte. Enfin, nous avons traité des ensembles et de la mesure des distances entre les documents, en utilisant les groupes linguistiques que nous avons proposés, ainsi que des approches basées sur des graphes.

Dans la première partie de la thèse, nous avons étudié les représentations de texte basées sur des graphes. Transformer le texte en graphiques n'est pas anodin et existait avant même que les mots incorporés ne soient introduits dans la communauté NLP. Dans notre travail, nous montrons que les représentations graphiques de texte peuvent capturer efficacement des relations telles que l'ordre, la sémantique ou la structure syntaxique. De plus, ils peuvent être créés rapidement tout en offrant une grande polyvalence pour de multiples tâches.

Dans la deuxième partie, nous sommes concentrés sur la régularisation structurée du texte. Les données textuelles souffrent du problème de dimensionnalité, créant de grands espaces de fonctionnalités. La régularisation est essentielle pour tout modèle d'apprentissage automatique, car elle permet de remédier au surajustement. Dans notre travail, nous présentons de nouvelles approches pour la régularisation de texte, en introduisant de nouveaux groupes de structures linguistiques et en concevant de nouveaux algorithmes.

Dans la dernière partie de la thèse, nous étudions de nouvelles méthodes pour mesurer la distance dans le mot englobant l'espace. Premièrement, nous présentons diverses méthodes pour améliorer la comparaison entre des documents constitués de vecteurs de mots. Ensuite, en présentant la comparaison des documents comme une correspondance bipartite pondérée, nous montrons comment nous pouvons apprendre des représentations cachées et améliorer les résultats pour la tâche de classification de texte.

Enfin, nous conclurons en résumant les principaux points de la contribution totale et en discutant des orientations futures.

PUBLICATIONS

The following publications and submissions under review are included in parts or in an extended version in this thesis:

1. Fragkiskos D. Malliaros and Konstantinos Skianis. "Graph-based term weighting for text categorization." In: *Proceedings of the 1st International Workshop on Data Science for Social Media and Risk, IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM. 2015, pp. 1473–1479.

2. Konstantinos Skianis, François Rousseau, and Michalis Vazirgiannis. "Regularizing Text Categorization with Clusters of Words." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1827–1837.

3. Konstantinos Skianis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. "Fusing Document, Collection and Label Graph-based Representations with Word Embeddings for Text Classification." In: *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12), NAACL*. 2018, pp. 49–58. (Best Paper Award)

4. Konstantinos Skianis, Nikolaos Tziortziotis, and Michalis Vazirgiannis. "Orthogonal Matching Pursuit for Text Classification." In: *Proceedings of the Fourth Workshop on User-generated Text (W-NUT), Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

5. Konstantinos Skianis, Fragkiskos D. Malliaros, Nikolaos Tziortziotis, and Michalis Vazirgiannis. "Boosting Tricks for Word Mover's Distance." Manuscript. 2019. (Submitted in ICWSM 2019)

6. Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. "Rep the Set: Neural Networks for Learning Set Representations." Manuscript. 2019. (Submitted in ICML 2019)

7. Konstantinos Skianis, Vlad Niculae, Guillaume Wisniewski, and Michalis Vazirgiannis. "Group Lasso for Linguistic Structured Attention." Manuscript. 2019. (Will be submitted in TACL 2019)

Furthermore, the following publications and submissions under review were part of my Ph.D. research but are not covered in this thesis, as they fall beyond the scope of the studied topics:

8. Antoine Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. "Gowvis: a web application for graph-of-words-based text visualization and summarization." In: *Proceedings of ACL System Demonstrations* (2016), pp. 151–156

9. Konstantinos Skianis, Maria-Evgenia G Rossi, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. "SPREADVIZ: Analytics and Visualization of Spreading Processes in Social Networks." In: *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1324–1327

10. Giannis Nikolentzos, Polykarpos Meladianos, Antoine Jean-Pierre Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. "Kernel graph convolutional neural networks." In: *International Conference on Artificial Neural Networks*. Springer, Cham. 2018, pp. 22–32

11. Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgianis. "GraKeL: A Graph Kernel Library in Python." In: *arXiv preprint arXiv:1806.02193* (2018)

12. Stamatis Outsios, Konstantinos Skianis, Polykarpos Meladianos, Christos Xypolopoulos, and Michalis Vazirgiannis. "Word Embeddings from Large-Scale Greek Web content." In: *Spoken Language Technology (SLT)*. 2018

Caminante, no hay camino, se hace camino al andar.
Wanderer, there is no path, the path is made by walking.

– Antonio Machado

ACKNOWLEDGMENTS

This dissertation constitutes the end of a long but amazing student trip. At this point, I would like to heartily thank the people that have supported me all those years.

In the three years I spent doing my PhD, I have been fortunate enough to interact with several people, both on a personal and a professional level. I believe that each one of these people has contributed in their own way towards the completion of this thesis. I am also sure that a few lines are not enough to acknowledge their contribution. Nonetheless, I will still make an attempt, hoping to include as many people as memory and time allows.

First and foremost, I would like to thank Prof. **Michalis Vazirgiannis**, without whom this all this academic trip, dating back to my bachelor years, would not have been possible. Prof. Vazirgiannis has been an excellent supervisor, with whom I engaged in numerous brainstorming sessions that helped me gain valuable knowledge and contribute to the field of natural language processing and machine learning. I have really admired his ability to direct me into the research directions that led to fruitful results while enjoying the freedom to experiment with ideas of my own. The joy and enthusiasm he has for research was contagious and motivational for me, even during tough times in the PhD pursuit. Professor Vazirgiannis treated me as an equal co-worker and respected my personality, while trying to make me better in all aspects, personal, academic and professional. He has been there not only as my supervisor, but also as a friend to whom I could trust anything.

I am also extremely grateful to my co-supervisor **Prof. Guillaume Wisniewski** who provided me with many insights in the area of regularization. I have learned a lot from working with him and interactions with him have always been lively.

Furthermore, I want to express my gratitude to the distinguished researchers who accepted to be part of my Ph.D. thesis defense committee: Particularly, I want to thank the members that reviewed my dissertation - - who gave detailed insightful comments about my PhD research work. The comments have helped me greatly in preparing this final version of my thesis.

I have been truly lucky to interact with many brilliant people at École Polytechnique. I would like to thank all the members of the DaSciM group, current and past, for all I learned from them and especially: Dr. Fragkiskos Malliaros, Dr. Nikolaos Tziortziotis, Dr. Giannis Nikolentzos, Dr. Francois Rousseau, Dr. Antoine Tixier, Dr. Maria Rossi, Dr. Polykarpos Meladianos, Dr. Panagiotis Korvesis, Stratis Limnios, Prof. Jesse Read, Christos Xypolopoulos and Dr. Christos Giatsidis.

During my academic years, I was extremely fortunate to have amazing collaborators (listed in chronological order of collaboration): Dr. Fragkiskos Malliaros, Dr. Francois Rousseau, Dr. Nikolaos Tziortziotis and Dr. Giannis Nikolentzos. I am grateful to all of them for the contributions presented in this dissertation and for the valuable knowledge I gained after various brainstormings.

I gratefully acknowledge École Polytechnique for providing the funding sources that made my PhD work possible.

On a personal level, there are so many people to which I want to express my most sincere gratefulness and affection, starting with my best friends Elias, Mos, Mpogamos, Giannis, Giorgos, Thanasis, Giorgos whom I've known since the age of five. Although there is a great distance between us, I could never imagine going through this journey without their unconditional love and everyday support.

Special credits go to my friends from my undergraduate studies, Giorgos, Kostas, Panos, Xaniw, Vasilis, Dimitris, Giorgos, Petros, Xaris, Petran, Stefanos. This work is dedicated especially to Thanasis, who left too early, but will always have a place in my heart.

I would also like to thank all the wonderful friends I have met in Paris. I would have never thought that I would spend 6 wonderful years in France and my friends did my staying here much more easier. Living at Fondation Hellenique in Cite Universitaire was an unforgettable experience that I would carry with me for the rest of my life. Fragkiskos, Stavros, Kostas, Giorgos, Filareti, Mitsos, Ifigeneia, Stergios, Stelios, Mairi, Stauros, Vaggelis, Maro, Cleo, Vardax, Nikos, Apostolis, Grigoris, Nikoletta made my life in Paris as well as my academic trip much easier and pleasant.

I would also like to warmly thank Phoebe for being in my life. Her support and care during the writing of this dissertation and beyond are immeasurable. Thank you for everything.

Last but not least, I would like to thank from the depths of my heart my family, Nikos, Euthalia and Giannis, as well as my grandparents Giannis, Maria, Kostas, Toula. Next my first cousins, Dimitris, Maria, Kostas, Vasilis, Thomas, Ektoras and Andrianos, my uncles and aunts, Ilias, Virginia, Annita, Vaggelis, Tasos, Liana, Dimitris, Katerina, Nikos, Sevi, Giorgos, Eleni and family friend Dora. They have been near me in this hard academic journey, especially during these challenging years of the PhD studies, and supported me in every decision I have made so far. This dissertation is dedicated to all of them. Thank you for everything you have done and continue doing for me.

Konstantinos Skianis
Paris, March 2019

To my family and friends
for their endless love and support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Text is undoubtedly the dominant method for storing, accessing and transferring knowledge since the start of human civilization. With the rise of the computer age, along with the invention of the Internet, text has been established as one of the most popular ways to access and distribute data between humans, sometimes preferred over speech (e.g. text messages over phone calls). Search engines, social networks and online purchase platforms are only some examples of the largest text-using technologies across the globe, creating a huge demand for fast and meaningful use of these data. The need for tools that process text effectively and efficiently is evident.

Modelling text for machines to understand is hard and thus a great research effort is done by the communities of Computational Linguistics (CL), Natural Language Processing (NLP) and Text Mining (TM). The complexity to model language for machines is thus high, as we humans have not even discovered how our brain models text and speech. Trying to mimic a more human-intelligent way, researchers exploited other techniques from areas like Machine Learning (ML) and more generally Artificial Intelligence (AI), in order to tackle the two challenging subparts of NLP, Natural Language Understanding (NLU) and Natural Language Generation (NLG). While great progress has been done on NLU, with NLG getting more attention during the years, traditional NLU tasks may return and require revisiting.

Living in the big data era, textual data are also increasing at a high rate. Humans can no longer handle simple tasks, for example even the process of classifying documents, due to the large volume. Through this dissertation we aim to battle such NLU problems by enhancing existing tools or developing novel techniques for better text representations, regularization in supervised learning scenarios and distances in word embeddings. As we will discuss later, our contribution is targeted, but not necessarily limited, for the task of text classification.

1.1 THESIS STATEMENT

This thesis contributes algorithms, tools, models and new insights to problems that arise in the area of natural language processing. We specifically:

- Develop methods for extracting rich graph-based representations of text, in order to feed supervised learning models.
- Extract meaningful linguistic structures, which we can use for regularization. We also design novel algorithms for standard and deep learning models.
- Extend and develop models that use distances between text documents.

Next, we provide an overview of the contributions of the dissertation with respect to the above points.

1.2 OVERVIEW OF CONTRIBUTIONS

Our research has been conducted in order to attack and address specific NLP tasks, which remain open to the community. In the following subsections, we will try to answer some questions that attracted our original interest and motivated us for our research contributions.

Graph-based Text Representations

How can we extract more meaningful representations than traditional ones?

Bag-of-Words (BoW) (Harris, 1954; Baeza-Yates and Ribeiro-Neto, 1999) had been extensively used in the past and is still used as a simple (and sometimes surprisingly effective) baseline. But this approach comes with losing a lot of information concerning the text.

Using the Graph-of-Words (GoW) (Mihalcea and Tarau, 2004; Rousseau and Vazirgiannis, 2013) approach, where we model a document or collection of words as a graph with nodes representing words and edges corresponding to co-occurence, we can harvest more information, for example, about word order. This graph representation enables us to extend its capabilities, since the standard version carries

as well limitations. By default, the Graph-of-Words approach gives the possibility to capture flows and relationships between the words of one document.

Building upon the Graph-of-Words, we introduce new ways of incorporating prior knowledge about the text structure in order to extract more meaningful graph-based representations. First, we propose the collection-level Graph-of-Words, in order to rank words via graph centralities, similarly to the Inverse Document Frequency (IDF). Next, in a supervised learning scenario, we develop label-level Graph-of-Words, so that we use the label information and create graphs by grouping all documents per available class. Last, we introduce word embeddings' similarities as weights in the Graph-of-Words.

Linguistic Structured Regularization

How can prior linguistic structures be used for regularizing text classification?

Since regularization was proposed for machine learning, researchers developed new methods for structured regularization. Assuming some prior grouping information is available, for example sentences or topics, one can utilize them in order to regularize a supervised learning model by using group structured regularization techniques (e.g. group lasso). A question that quickly arises is how do different groups affect text classification with group lasso?

First, we show that groups by clustering in the word embedding space, either by community detection in the Graph-of-Words, or by Latent Semantic Indexing can be as effective and efficient as state-of-the-art linguistic structured regularizers. Furthermore our suggested schemes achieve big sparsity, without losing significant accuracy, as well as faster learning times.

Last, new ways for structured regularization can be explored, diverging from group lasso variants. Our proposal is the Overlapping Orthogonal Matching Pursuit algorithm, a greedy group feature selection method. With this approach we show that we can achieve sparser models almost in all test cases, with a very high accuracy.

Boosting and Learning Distances in Text

How can we boost existing approaches for measuring distances using word embeddings or design new methods to learn these distances?

Assuming we have two documents of text and we want to measure similarity between them, we need methods like the Word Mover's Distance (WMD). In order to boost this existing technique one can analyze and tune its components. More specifically, we worked on three of these components: a) stopword removal, so that we check if different stopwords can affect the document distance; b) cross document-topic comparison, by augmenting the documents with nearest words that may belong to the same topic and c) convex metric learning, where we replace the non-convex part of the Supervised Word Mover's Distance.

Last, we introduce a novel neural network architecture based on set matching which can learn sets for comparison purposes. In the supervised learning scenario We show that our simple method performs comparable to state-of-the-art methods for the task of text classification.

1.3 SOFTWARE AND LIBRARIES

Most of the code pertaining to the projects presented in this dissertation has been developed in two programming languages: Python 2.7, 3.5 and Matlab.

For the graph representation and computation in Python, we used the most popular libraries, networkx (Hagberg et al., 2008), along with igraph (Csardi and Nepusz, 2006) and graph-tool (Peixoto, 2014) for fast operations.

Working on the text classification task, for the standard classifiers, we used the Python scikit-learn library (Pedregosa et al., 2011), as well as Matlab toolboxes. For the deep learning baselines as well as our contributed approaches, we used Keras (Chollet, 2015) with Tensorflow (Abadi et al., 2016) and PyTorch (Paszke et al., 2017).

1.4 OUTLINE OF THE THESIS

The rest of the dissertation is organized as follows. In Chapter 2 we present basic concepts and background material that will be used throughout the thesis. In

Chapter 3 we present novel methods for graph-based representations of text. Next, Chapter 4 shows new linguistic groups that can be used for regularization, as well as new methods for structured regularization in standard and deep learning architectures. In Chapter 5 we present our work towards enhancing Word Mover's Distance and creating new models to learn distances via weighted bipartite matching and graph flows. Finally, in Chapter 6, we offer concluding remarks about the topics covered in the dissertation and future research directions.

2

BASIC CONCEPT AND PRELIMINARIES

In this chapter, we describe the foundations upon which we conducted our research. Since we were essentially interested in mining and extracting information from textual content, i.e. tackling the field of Text Mining (TM), we show our focused contribution under an application interpretation. More specifically, we describe the full framework of the single-label multi-class text classification task and show where our methods can be fitted as components.

2.1 STANDARD TEXT REPRESENTATION

By default, current computer machines fail to understand natural language in a human way, due to their ability to handle only real and binary values. It is widely acceptable that we are far from making machines comprehend natural language in its physical form. In order to proceed, and because of the continuously growing volume of data, we came up with simple and fast ways of representing text as machine readable information.

When we refer to a dataset, we mean that a collection of documents, denoted by $\mathcal{D}$, is given to us, where one label y is assigned to each one of these documents. A document, referred as well as point or instance, carries a specific amount of raw text. It can be a web page, social network post, an email or review. A document d or x consists thus of a sequence of words:

$$d = (t_1, t_2, \ldots, t_{|d|}) \tag{2.1}$$

where t_i represents a word. The collection of distinct words form the vocabulary T (or dictionary, or lexicon) of the given dataset. In this dissertation, converting a document into a sequence of words, known as tokenization, is assumed to be solved. Last, for test cases, we use datasets written in modern English.

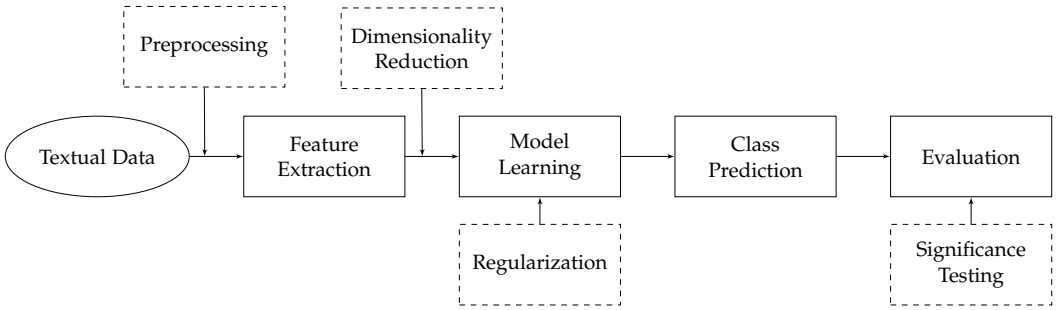


Figure 2.1: Basic pipeline of the Text Classification task.

2.2 SINGLE-LABEL MULTI-CLASS CATEGORIZATION

Text Categorization (TC), a.k.a. text classification, is defined as the task of automatically predicting the class label, a.k.a. category of a given input textual document. It finds applications in a wide variety of domains, from news filtering and document organization to opinion mining and spam filtering.

In this dissertation, we only consider the case for which we want to predict a single label per document but not necessarily restricted to a binary choice – hence the "single-label multi-class" denomination (multi-label text categorization may also be referred to as topic spotting where we want to predict probability weights for each pre-defined topic).

Until recently, compared to other application domains of the general machine learning task of classification, TC's specificity lied in its high number of features, its sparse feature vectors, its multi-class scenario and its skewed category distribution. For instance, when dealing with collections of thousands of news articles, it is not uncommon to have millions of n-gram features, only a few hundreds actually present in each document, tens of class labels – some of them with thousands of articles and some others will only a few hundreds. These particularities have to be taken into account when considering feature selection, learning algorithms and evaluation metrics as well as alternative document representations like in our research.

Figure 2.1 depicts the basic pipeline of a text classification system. With the ellipses shape we represent our input, the textual data, with blocks we indicate the processes and with dashed borders the ones that can be seen as auxiliary components. The pipeline that typically is followed to deal with the problem is

similar to the one applied in any classification problem; the goal is to learn the parameters of a classifier from a collection of training documents (with known class information) and then to predict the class of unlabeled documents. As we observe the pipeline consists of multiple components.

These components are preprocessing, feature extraction, dimensionality reduction, model learning, regularization, class prediction and evaluation. Some of these are not necessary parts of the text classification task, and thus can be removed. For example, dimensionality reduction can be omitted if the number of terms is not very high. Similarly, regularization is an additional component for the model learning process and consequently can be selected optionally.

From all the text classification components, our contribution is focused on the feature extraction, model learning and regularization parts. The graph-based text representations in Chapter 3 can be seen as a feature extraction component. Next, the linguistic structured regularization methods in Chapter 4 can be seen as an optional component for the model learning process, belonging obviously in the regularization part. In the last part, the first work for boosting distances in word embeddings from Chapter 5, can also be interpreted as a feature extraction part, while the second part with learning sets can be seen as a model learning component.

2.2.1 *Feature Extraction*

The first step in text categorization is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task. The main approach here is apply the Vector Space Model (VSM), a spatial representation of text documents. In this model, each document is represented by a vector in a n-dimensional space, where each dimension corresponds to a term (i.e., word) from the overall vocabulary of the given document collection. More formally, let $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ denote a collection of m documents, and $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be the dictionary, i.e., the set of terms in the corpus $\mathcal{D}$. The set of terms $\mathcal{T}$ can be obtained either directly from the documents or after applying some standard natural language processing techniques, such as tokenization, stop-words removal and stemming (Baeza-Yates and Ribeiro-Neto, 1999). Each document $d_i \in \mathcal{D}$ is represented as a vector of term weights $d_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,n}\}$, where $w_{i,k}$ is the weight of term k in document d_i.

That way, data can be represented by the Document-Term matrix of size $m \times n$, where the rows correspond to documents and the columns to the different terms (i.e., features) of set $\mathcal{T}$. Note that, due to the large number of features of this matrix, dimensionality reduction techniques, such as Latent Semantic Analysis (LSA) (Yu et al., 2008), can be applied (Sebastiani, 2002).

2.2.2 *Classifiers*

Most considered approaches were first used with unigrams as features and then extended to n-grams. A plethora of classifiers are available to train on these features. We note in particular the seminal work of Joachims (1998) who was one of the first to propose the use of a linear Support Vector Machine (SVM), a geometric classifier, with TF-IDF unigram features for the task. This approach is one of the standard baselines because of its simplicity yet effectiveness (unsupervised n-gram feature mining followed by standard supervised learning). Logistic Regression (LR) and Linear Least Square Fit (LLSF) have been also used for text classification; we still note the works of Zhang and Oles (2001), and Genkin et al. (2007). Another popular approach is the use of Naive Bayes (NB), a probabilistic classifier, and its multiple variants (McCallum and Nigam, 1998), in particular for the subtask of spam filtering (Androutsopoulos et al., 2000). Finally, k-Nearest Neighbors (kNN), an example-based classifier, has also been considered in the literature (Creecy et al., 1992; Yang, 1994; Larkey and Croft, 1996).

More recently, deep learning has started to be used for solving the task as well (Sarikaya et al., 2011). After the appearance of the word embeddings (Mikolov et al., 2013a), more deep learning architectures were presented (Kim, 2014), to achieve state-of-the-art results.

2.2.3 *Regularization*

Regularization is a tool to constrain/regularize or shrink the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.

Regularization, significantly reduces the variance of the model, without substantial increase in its bias. Its most important tuning parameter λ, used in the

regularization techniques, controls the impact on bias and variance. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance. Till a point, this increase in λ is beneficial as it is only reducing the variance (hence avoiding overfitting), without losing any important properties in the data. But after a certain value, the model starts loosing important properties, giving rise to significant bias in the model and thus underfitting. Therefore, the value of λ should be carefully selected. Lastly, regularization is also associated to sparsity, using fewer weights, and thus model interpretability.

2.3 EVALUATION

As we want to measure how good the performance of a model is, we require evaluation as a standard process, after the classification task. We assume a classification task for which we have ground truth data, a.k.a. a gold standard for the expected outcome. For instance, it could be the set of true class labels in text classification. In practice, a lot of these tasks have binary outcomes (e.g. spam/non-spam or positive/negative review) and we refer to one of the two outcomes as the "positive", usually the one corresponding to the task (the other alternative being the default or the "negative" one). We may also have multi-class cases, where one can consider one class as positive and the rest as negative and then average the metrics as we will see later on.

In this context, given a binary classification task with positive and negative examples or even a multi-class one, we want to evaluate how well a system manages to predict accurately. There are four types of possible predictions (Swets, 1963):

1. True Positive (TP) – the system correctly predicts a positive class for a positive example, resulting in a correct acceptance
2. True Negative (TN) – the system correctly predicts a negative class for a negative example, resulting in a correct rejection
3. False Positive (FP) – the system wrongly predicts a positive class for a negative example, resulting in a false alarm
4. False Negative (FN) – the system wrongly predicts a negative class for a positive example, resulting in a miss

Now, using the aforementioned rations, we can formulate the metrics which we will use for the evaluation process.

ACCURACY An intuitive way of measuring a system's performance would be to count the number of successful predictions and then divide them by the total number of predictions. This is given by the accuracy metric (Acc) and is defined as follows:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.2}$$

Although accuracy is one of the most popular metrics for text classification, as well as other tasks of ML, it could be misleading. If there is high imbalance in the class distribution, even a majority class classifier (classifying all instances into the large class) will have very high accuracy score, misleading us to believe that performance is good.

PRECISION is basically the fraction of correct predictions restricted to the positive class (which alleviates the pitfall of accuracy when the negative examples belong to the majority class). The definition of the Precision metric (P):

$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.3}$$

For example, in spam filtering, a false alarm is detrimental since it means tagging a real email as junk and therefore hiding it from the user, which translates into wanting to reduce FP w.r.t. TP, thus increasing Precision.

RECALL Conversely, consider a medical doctor that uses a specialized search engine to retrieve all patients with cancer – we realize that no miss can be tolerated, which translates into wanting to reduce FN w.r.t. TP, leading to the definition of a metric known as Recall (R), a.k.a. sensitivity:

$$\text{R} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.4}$$

F-MEASURE AND F1-SCORE As aforementioned, depending on the task at hand and the context, sometimes the sole precision matters or only the recall is of interest but in general both precision and recall are important. However, it is hard to compare two systems where one has a better precision and the other one a better

recall. Therefore, the research community devised a combined measure of precision and recall. Van Rijsbergen (1979) proposed a measure of effectiveness E to capture the trade-off between precision and recall and it is actually its complement F(= 1 − E) that became popular, known as F-measure (F_β):

$$F_\beta = \frac{1}{\alpha/P + (1-\alpha)/R} = (1+\beta^2)\frac{P \times R}{\beta^2 P + R}, \alpha = \frac{1}{1+\beta^2} \tag{2.5}$$

α controls the relative weight we want to give to precision w. r. t. recall. When we consider that precision and recall are of equal importance (i. e. $\alpha = 1/2$, $\beta = 1$), the metric is usually referred to as F1-score (F1):

$$F1 = \frac{2P \times R}{P + R} \tag{2.6}$$

which corresponds to the harmonic mean of precision and recall. Note that if the precision and recall are equal then they are also equal to the F1-score.

MICRO- VS. MACRO-AVERAGED METRICS One can consider micro- and macro-averaged metrics, which means that when comparing two systems, we can perform a pairwise comparison of respectively the per-example binary decisions and the per-class/per-task metrics. Per-example binary decisions (e.g. spam/non-spam) can only tell us if the two systems differ and if so, which one was right. Per-class/per-task metric (e.g., F1-score for a given category) can also help us quantify the magnitude of the difference.

SIGNIFICANCE TESTING However, when evaluation measures are averaged over a number of classes or tasks, we can obtain an estimate of the error with that measure and statistical significance testing becomes applicable. Intuitively, the decision to consider an improvement significant is strengthened (1) when the difference values are relatively high; or (2) when these values are, more or less, always in favor of one system; and (3) when the sample size grows.

NULL HYPOTHESIS The preliminary assumption, or null hypothesis Ho, is that the systems are equivalent in terms of performances (i.e. the observed improvement is due to chance). The significance test will attempt to disprove this hypothesis by determining a p-value, i.e. a measurement of the probability that the observed improvement could have occurred assuming that Ho holds. Under Ho,

each test computes a statistic T and calculates the achieved significance level of this test, which is the probability of observing a value at least as extreme as T when Ho holds: $p = P(X \geqslant T)$ where X is the random variable for the statistic and P the assumed probability distribution, typically the standard normal distribution. If this probability is less than a pre-defined significance level α (typically 0.05 or 0.01 to be more conservative), which corresponds to the Type I error rate we are willing to accept, we may reject the null hypothesis with $1 - \alpha$ (i. e. 95% or 99%) confidence and conclude that the two systems are significantly different, i. e. that the alternative hypothesis is at least more likely than the null hypothesis. The effect size on the other hand measures the magnitude of the improvement and relates in that sense more to the idea of noticeable and material changes (Sparck-Jones and Bates, 1977).

DISTRIBUTION ASSUMPTIONS Significance tests fall into a number of different categories, in particular parametric vs. non-parametric (a.k.a. distribution-free) depending on whether we make specific assumptions about the distribution of the measurements and their errors. We considered in our experiments the sign test (Conover and Conover, 1980). Paired tests (as opposed to independent tests) are the most suitable for comparing values produced by two systems for the same set of independent observations (in our case queries or documents at the macro-level and classification decisions at the micro-level) and also because there is no risk of "contamination" between the two automated systems (as opposed to let's say having each patient taking the drug and the placebo). These tests are by far the most used in text categorization (Yang and Liu, 1999).

SIGN TEST VS. T-TEST The sign test* looks only at which system performed better: if one system performs better than the other far more frequently than would be expected on average, then this is strong evidence that it is superior. The sign test can be used for both micro and macro-averaged metrics. The Student's t-test† compares the magnitude of the difference between systems to the variation among the differences. If the average difference is large compared to its standard error, then the systems are significantly different. The t-test assumes that the difference follows the normal distribution, but it often performs well even when this assumption is violated (Hull, 1993; Yang and Liu, 1999). Since we are considering both

* https://stat.ethz.ch/R-manual/R-devel/library/stats/html/binom.test.html
† https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

binary classification and multiclass decisions and the t-test only makes sense for macro-averaged metrics where we can measure the magnitude of the difference between real-valued metrics, we end up using sign test across all our experiments.

3

GRAPH-BASED REPRESENTATIONS OF TEXT

In this chapter we provide a novel framework for graph-based text representations. Our work proposes new ways of transforming textual data to rich graph representations that can capture more information than traditional approaches. First, we present related work about standard techniques based in Bag-of-Words and existing ones based on graphs, along with the required background. Next, we present novel methods for graph-based representations of text. Last, we show an extensive experimental study for the text categorization task. We provide all code and data online*.

3.1 INTRODUCTION

Contrary to the traditional Bag-of-Words approach, we consider the Graph-of-Words (GoW) model in which each document is represented by a graph that encodes relationships between the different terms. Based on this formulation, the importance of a term is determined by weighting the corresponding node in the document, collection and label graphs, using node centrality criteria. We also introduce novel graph-based weighting schemes by enriching graphs with word-embedding similarities, in order to reward or penalize semantic relationships. Our methods produce more discriminative feature weights for text categorization, outperforming existing frequency-based criteria.

With the rapid growth of the social media and networking platforms, the available textual resources have been increased. Being able to automatically analyze and extract useful information from textual data is an important task with many applications. Text categorization or classification (TC) refers to the supervised learning task of assigning a document to a set of two or more predefined categories (or classes) (Sebastiani, 2002). TC can be applied in several domains. A well-known application is the one of sentiment analysis, where the goal is to identify subjective information from text corpora. Other popular applications of TC

* github.com/y3nko/Graph-Based-TC

include spam detection (Androutsopoulos et al., 2000), news filtering (Aggarwal and Zhai, 2012), as well as novel ones like computational phenotyping (Zeng et al., 2018).

In the TC pipeline, each document is modelled using the so-called Vector Space Model (VSM) (Baeza-Yates and Ribeiro-Neto, 1999). The main issue here is how to find appropriate weights regarding the importance of each term in a document. Typically, the Bag-of-Words (BoW) model is applied and a document is represented as a multiset of its terms, disregarding co-occurence between the terms; using this model, the importance of a term in a document is mainly determined by the frequency of the term.

Although several variants and extensions of this modeling approach have been proposed (e.g., the n-gram model (Baeza-Yates and Ribeiro-Neto, 1999)), the main weakness comes from the underlying term independence assumption, where the order of the terms is also completely disregarded.

After the introduction of deep learning models for text classification (Blunsom et al., 2014; Kim, 2014), recent work by Johnson and Zhang (2015) shows how we could effectively use the order of words with Convolutional Neural Networks (CNN) (LeCun and Bengio, 1995). In many cases though, space and time limitations may arise due to complex neural network architectures. As stated in work by Joulin et al. (2017), computation can still be expensive and prohibitive. Moreover, word vectors, unless pre-trained, require also a considerable time to learn.

In this chapter, we explore fast graph-based term weighting criteria for text classification that go beyond the term independence assumption. Under this formulation, we create graph representations of collections of words, with nodes representing the terms and edges capturing co-occurrence relationships of terms.

The basic advantage of our approach is that we are able to augment the unigram feature space of the learning task with weights that implicitly consider information about n-grams in the document as well as the collection of documents – expressed by paths in the graph – without increasing the dimensionality of the problem. Furthermore, we introduce word-embedding similarities as weights in the GoW approach, in order to further boost the performance of our methods. Finally, we successfully mix document, collection and label GoWs along with word vector similarities into a single powerful graph-based framework.

The main contributions of our work can be summarized as follows:

- *Graph-based term weighting schemes*: adopt a graph-based representation of documents and derive novel term weighting schemes for TC through a ranking process in the interconnected feature space defined by the graph.
- *Inverse Collection Weight (ICW)*: propose a novel graph-based, term weighting criterion to penalize the importance of terms across the collection level.
- *Label Weight (LW)*: a novel graph-based, term weighting criterion to reward the importance of terms across class graphs level.
- *Word embeddings*: we utilize word vectors as semantic information and integrate similarities between words to our graph-based approach as weights in the edges.
- *Empirical study*: we perform experiments on well-known datasets for TC. Our results indicate that the proposed weighting schemes are able to outperform existing frequency-based ones.

3.2 RELATED WORK

In this section we review the related work, which can be placed into four main categories: term weighting for document representation, text categorization, deep learning for text categorization, graph-based text classification and graph-based methods in text mining, natural language processing and information retrieval.

TERM WEIGHTING SCHEMES A core aspect in the Vector Space Model for document representation, is how to determine the importance of a term within a document. This is central, and still active, research topic that goes back to the origins of IR; since then, many criteria have been introduced with the most prominent ones being TF, TF-IDF (Salton and Buckley, 1988; Singhal et al., 1996; Baeza-Yates and Ribeiro-Neto, 1999; Robertson, 2004) and Okapi BM25 (Robertson et al., 1996), while some recent ones include N-gram IDF (Shirakawa et al., 2015). An extensive study can be found in Manning et al. (2008).

With the advances on learning techniques for textual data, many of these weighting schemes were considered or extended in order to deal with supervised and

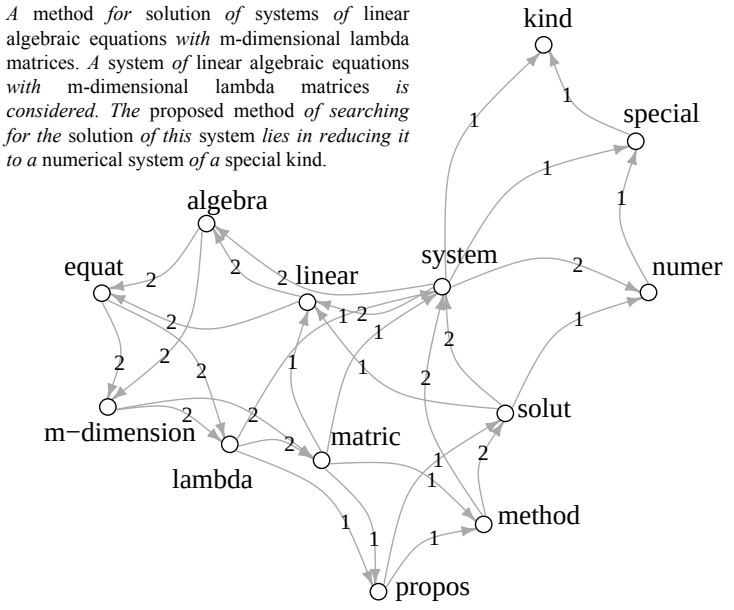


Figure 3.1: A Graph-of-words example.

unsupervised tasks. Some examples include the Term Frequency - Inverse Corpus Frequency (TF-ICF) (Reed et al., 2006) method proposed for document clustering and the TF-RF scheme (Lan et al., 2009) used in text categorization. Lan et al. (2005) conducted a comparative study of frequency-based term weighting criteria for text categorization; one of their outcomes was that, in many cases, the IDF factor is not significant for the categorization task, leading to no improvement of the performance. It is interesting to point out here that, more specialized approaches have been proposed for specific classification tasks, such as the Delta TF-IDF method that constitutes an extension of TF-IDF for sentiment analysis (Martineau and Finin, 2009).

However, most of the previously proposed frequency-based weights consider the document as a Bag-of-Words; that way, any structural information about the ordering or in general, syntactic relationship of the terms, is ignored by the weighting process. The notion of dependencies between terms is introduced via a Graph-of-Words (GoW) representation model (Rousseau and Vazirgiannis, 2013). Under this model, each term is represented as a node in the graph and the edges capture co-occurrence relationships of terms with a specified distance in the document. We present a toy example of a Graph-of-Words in Figure 3.1.

TEXT CATEGORIZATION A number of diverse approaches have been proposed for TC (Joachims, 1998; McCallum and Nigam, 1998; Nigam et al., 2000; Sebastiani, 2002; Kim et al., 2006a). The first step of TC concerns the feature extraction task, i.e., which features will be used to represent the textual content. Typically, the straightforward Bag-of-Words approach is adopted, where every document is represented by a feature vector that contains boolean or weighted representation of unigrams or n-grams in general. In the case of weighted feature vectors, various term weighting schemes have been used, with the most well-known ones being TF (Term Frequency), TF-IDF (Term Frequency - Inverse Document Frequency).

Although these weighting schemes were initially introduced in the NLP and IR fields, they have also been applied in the text classification task. Paltoglou and Thelwall (2010) reported that, in the case of sentiment analysis, extensions of the TF-IDF weighting schemes introduced in the IR field, can further improve the classification accuracy. A comprehensive review of this area is offered in the article by Sebastiani (2002).

DEEP LEARNING FOR TC With the rise of deep learning models, CNN architectures were applied for text classification (Blunsom et al., 2014; Kim, 2014), presenting state-of-the-art results for many datasets. Work by Johnson and Zhang, 2015 was the first to exploit the order of the words in deep neural network architectures. Next, Zhang et al. (2015) presented character-level CNNs for the task of TC, with Conneau et al. (2017) extending that to much deeper models. Johnson and Zhang (2016) enabled LSTMs for supervised and semi-supervised text categorization, where text regions of variable and large sizes can be embedded. Later, Johnson and Zhang (2017) introduced a deep pyramid CNN, a low-complexity word-level deep convolutional network that can efficiently represent long-range associations in text. Joulin et al. (2017) proposed a novel text classifier based on a shallow neural network which achieves equivalent performance to state-of-the-art TC models, with faster learning times, able to run in a standard multicore CPU. Last, Miyato et al. (2017) extended adversarial and virtual adversarial training to the text domain by applying perturbations to the word embeddings in a recurrent neural network rather than to the original input itself. Our work is not considered a full model that includes the feature extraction and learning components, as the aforementioned methods, but is focused only on the extraction part.

GRAPH-BASED TEXT CATEGORIZATION In the related literature, most of the graph-based method for TC, rely on graph mining algorithms that are applied to extract frequent subgraphs, which are then used to produce feature vectors for classification (Deshpande et al., 2005; Jiang et al., 2010; Rousseau et al., 2015; Nikolentzos et al., 2017b). The basic shortcoming of those methods stems from the computational complexity of the frequent subgraph mining algorithm. Furthermore, most of these methods require from the user to set the support parameter, which concerns the frequency of appearance of a subgraph. Wang et al. (2005) introduced a term graph model that, contrary to our approach, captures the relationships among terms using frequent itemset mining algorithms. Aery and Chakravarthy (2005) proposed InfoSift, a graph-matching based method for document classification. Lately, deep learning has been combined with graph-based approaches for the task of text classification. Peng et al. (2018) proposed a graph-CNN based deep learning model to first convert texts to Graph-of-Words, and then use graph convolution operations for the convolution operation. Yao et al. (2019) build a single text graph for a corpus based on word co-occurrence and document word relations. Then a Text Graph Convolutional Network (Text GCN) is learned for the corpus.

Close to our work are the approaches followed by Hassan et al. (2007) and Malliaros and Skianis (2015); they explored how random walks and other graph centrality criteria can be applied to determine the importance of a term.

The basic difference of our work is that: (i) we consider the concept of collection-level graph which leads to the Inverse Collection Weight (ICW) for penalizing globally frequent terms; (ii) we also propose a simple-yet-effective mechanism to enhance the graph-based document representation with word-embeddings, by selecting appropriate weights for the edges of the Graph-of-Words – deriving novel term weighting schemes. In this work, we revisit and further extend this approach, showing, among other things, that we can rely on simpler and easier to compute graph-based criteria – such as the degree of a node – to achieve even better classification results.

GRAPH-BASED NLP & IR Representing text documents as graphs is a well-known approach in NLP and IR. Dhillon (2001) proposed to deal with the document clustering problem using a bipartite graph model. The first partition of the graph corresponds to the terms of the collection, while the other to the documents;

then, the existence of an edge indicates the appearance of the term in the document. Then, graph partitioning algorithms can be used to solve the document clustering problem. Still, one question here is how to weight the edges of the graph and in most of the related work, the TF-IDF scheme is adopted.

Related methods are proposed for keyword extraction (KE) and ad hoc IR. TextRank algorithm, proposed by Mihalcea and Tarau (2004), was among the first works that considered a random walk model similar to PageRank, over a graph representation of the document, in order to extract representative keywords and sentences. Later, several methods for those tasks were followed (Erkan and Radev, 2004; Litvak and Last, 2008; Boudin, 2013; Lahiri et al., 2014; Rousseau and Vazirgiannis, 2015).

Another domain where graph-based term weighting schemes have been applied is the one of ad hoc Information Retrieval (Rousseau and Vazirgiannis, 2013). An interesting survey can be found in the work of Blanco and Lioma (2012) for a detailed description of graph-based methods in the text domain.

3.3 PRELIMINARIES AND BACKGROUND

In this section, we briefly discuss the basic formulation of the TC problem, as well as the frequency-based weighting criteria that are derived from the traditional BoW model, namely TF and TF-IDF. Then, we introduce the graph-theoretic concepts upon which our framework for TC is built.

3.3.1 *TC Pipeline in the BoW Model*

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ be a collection of documents and let $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ be the set of predefined categories. Text categorization is considered the task of assigning a boolean value to each pair $(d_i, c_i) \in \mathcal{D} \times \mathcal{C}$, i.e., assigning each document to one or more categories (Sebastiani, 2002). The main point here is how to find appropriate weights for the terms within a document. As we will present below shortly, our approach utilizes network centrality criteria which are briefly summarized below.

The first step in the TC process is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and

the classification task. Here, we consider the widely used Vector Space Model, i.e., the spatial representation in which each document is represented by a vector in the n-dimensional space defined by the terms $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ of the overall vocabulary of the collection (Baeza-Yates and Ribeiro-Neto, 1999). That way, each document $d_i \in \mathcal{D}$ is represented by a vector of weights $d_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,n}\}$, where $w_{i,k}$ is the weight of term k in document d_i.

In the traditional Bag-of-Words model, each document is represented as the bag (multiset) of its words, disregarding the ordering or in general, any potential dependencies between the terms of the document. Under this model, the importance of a term in a document is mainly determined by the frequency of the term. That is, the weight of a term $t \in \mathcal{T}$ within a document $d \in \mathcal{D}$ is based on the frequency $tf(t, d)$ of the term in the document (TF weighting scheme). Furthermore, terms that occur frequently in one document but rarely in the rest of the documents, are more likely to be relevant to the topic of the document. This is known as the inverse document frequency (IDF) factor, and is computed at the collection level. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient, as follows:

$$\text{IDF}(t, \mathcal{D}) = \log\left(\frac{m+1}{|\{d \in \mathcal{D} : t \in d|\}}\right),$$

where m is the total number of documents in collection $\mathcal{D}$, and the denominator captures the number of documents that term t appears. Then, the TF-IDF scheme is produced by the multiplication of the TF and IDF factors.

One of the most commonly used TF-IDF weighting scheme, proposed by Singhal et al. (1996), is called pivoted normalization weighting, where we also take under consideration the length of the document, as well as the average document length in the training dataset.

$$\text{TF-IDF}(t, d) = \frac{1 + \ln\left(1 + \ln(\text{TF}(t, d))\right)}{1 - b + b \times \frac{|d|}{\ell}} \times \text{IDF}(t, \mathcal{D}), \tag{3.1}$$

where d is the length of the document, ℓ is the average document length and parameter b is set by default to 0.20 (as suggested in (Singhal et al., 1996)). The TF-IDF scoring function captures the intuitions that (i) the frequency of a term in a document is proportional of how representative it is for its content, and (ii) the higher the number of documents a term occurs in, the less discriminating it is

(term specificity)[†]. In this chapter, we use a simple version of TF-IDF, which gave the best results. This weighting scheme (and its variants) has been widely used in the TC task; as we will present shortly, the TF and TF-IDF weighting mechanisms will be the main baseline approaches for our experimental evaluation.

As we understand, the main point here is how to find appropriate weights for the terms within a document, but based on a graph representation.

3.3.2 *Node Centrality Criteria*

Centrality[‡] represents a central notion in graph theory and network analysis in general; it consists of measures that capture the relative importance of the node in the graph based on specific criteria (Newman, 2010).

Local centrality criteria. One important characteristic of the centrality measures is that they consider either local information of the graph (e.g., degree centrality, in-degree/out-degree centrality in directed networks, weighted degree in weighted graphs, clustering coefficient) (Newman, 2010), or more global information – in the sense that the importance of a node is determined by the properties of the node globally in the graph (e.g., PageRank, closeness). Let $G = (V, E)$ be a graph (directed or undirected), and let $|V|$, $|E|$ be the number of nodes and edges respectively. Next, we define basic centrality criteria that are used in the proposed methodology.

Degree centrality. The degree centrality is one of the simplest local node importance criteria, which captures the number of neighbors that each node has. Let $\mathcal{N}(i)$ be the set of nodes connected to node i. Then, the degree centrality can be derived based on the following formula:

$$\text{degree\_centrality}(i) = \frac{|\mathcal{N}(i)|}{|V|-1}.$$

In-degree & out-degree centrality. These centralities constitute extensions of the degree centrality in directed networks, where we treat independently the in-degree

† Several variants of the TF-IDF score have been proposed. See also the description given in Sebastiani (2002).

‡ en.wikipedia.org/wiki/Centrality.

(number of incoming edges) and out-degree (number of outgoing edges) of each node.

Global centrality criteria. The closeness and pagerank centrality could be also used as global centrality criteria, in order to capture important nodes using global information of the graph.

Closeness centrality. Let $\mathtt{dist}(i, j)$ be the shortest path distance between nodes i and j. The closeness centrality of a node i is defined as the inverse of the average shortest path distance from the node to any other node in the graph (Newman, 2010):

$$\text{closeness}(i) = \frac{|V|-1}{\sum_{j \in V} \mathtt{dist}(i, j)}.$$

Contrary to degree centrality, the closeness score is a global metric, in the sense that it combines information from all the nodes of the graph. Here we compute the closeness centrality in the undirected graph.

PageRank centrality. PageRank counts the number and quality of edges to a node to determine a rough estimate of how important the node is:

$$\text{PR}(i) = \frac{1-\alpha}{|V|} + \alpha \sum_{\forall (j,i) \in E} \frac{\text{PR}(j)}{\text{out-deg}(j)},$$

where α is the damping factor and out-deg(i) denotes the out degree on node i.

3.4 PROPOSED FRAMEWORK

In this section, we present the components of the proposed graph-based framework for TC. We adopt the Graph-of-Words document representation, where documents are represented as graphs that capture term co-occurrence relationships within a fixed-size sliding window. In this section, we show (i) how to derive meaningful term weighting schemes for TC – at document, collection and class level – and (ii) how to introduce similarities of terms in the word-embedding space as weights in the graphs.

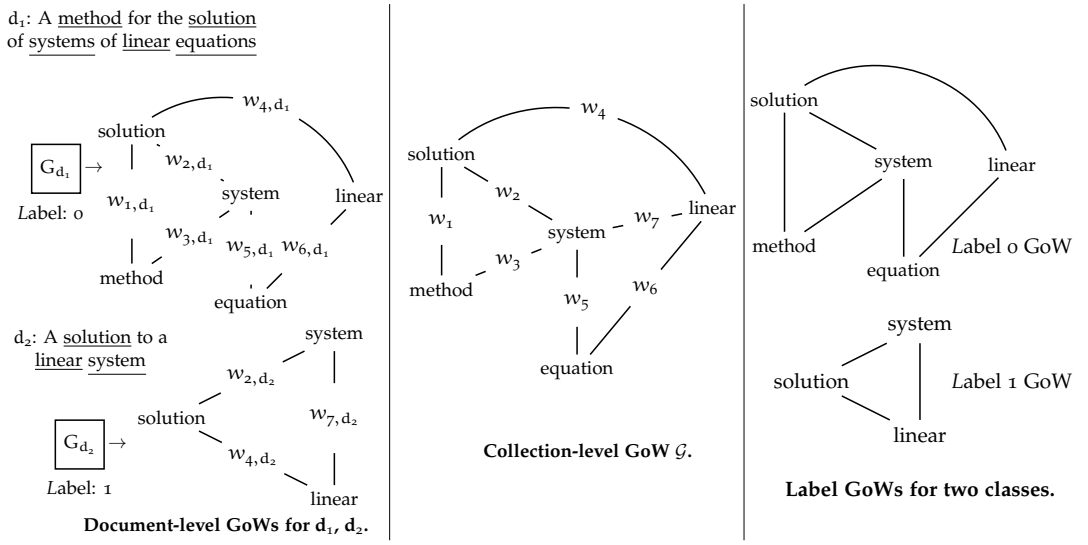


Figure 3.2: Example of document, collection-level and label GoWs for a collection composed by two documents and window size $w = 3$. Weights on the edges of G_{d_1} and G_{d_2} correspond to the similarity of two terms in the vector space. Here, Label GoWs are the same with Document GoWs (one document per class-label).

3.4.1 *Graph Construction*

We model documents as graphs that capture dependencies between terms. More precisely, each document $d \in \mathcal{D}$ is represented by a graph $G_d = (V_d, E_d)$, where the nodes correspond to the terms t of the document and the edges capture co-occurrence relationships between terms within a fixed-size sliding window of size w. That is, for all the terms that co-occur within the window, we add edges between the corresponding nodes of the graph. Note that, the windows are overlapping starting from the first term of the document; at each step, we simply remove the first term of the window and add the new one from the document. Figure 3.2 gives a toy example of the construction of GoW for a collection composed by two documents. As graphs constitute rich modelling structures, several parameters about the construction phase need to be specified, including the directionality of the edges, the addition of edge weights, well as the size w of the sliding window.

- **Directed vs. undirected graph.** One parameter of the model is if the graph representation of the document will be directed or undirected (Easley and Kleinberg, 2010). Directed graphs are able to preserve actual flow on a text, while in undirected ones, an edge captures co-occurrence of two terms what-

ever the respective order between them is. We have tested both choices, observing that undirected graphs perform significantly better; it becomes evident that the actual ordering of terms is not a discriminative factor in TC.

- **Weighted vs. unweighted graph.** One approach is to consider weighted graphs. That is, the higher the number of co-occurrences of two terms in the document, the higher the weight of the corresponding edge (i.e., the weight of each edge will be equal to the number of co-occurrences of its endpoints). The second and more simple option, is to consider unweighted graphs.

- **Size w of the sliding window.** In the construction of the graph, we add edges between the terms of the document that co-occur within a sliding window of size w. Although by increasing the size of the window we are able to capture co-occurrence relationships between not necessarily nearby terms (similar to the notion of (long) n-grams), the produced graph becomes relatively dense. From our experimental results, we have observed that small window sizes give persistently better classification results.

To summarize, the key point of the graph-based representation for TC is the fact that it deals with the term independence assumption. Even if we consider the n-gram model, still information about the relationship between two different n-grams is not fully captured – as happens in the case of graphs. This has also been noted in other application domains (e.g., IR (Rousseau and Vazirgiannis, 2013)).

Note that, the above procedure, as has been applied in our framework, concerns unigrams; we consider that a similar approach can potentially be applied to build a graph using n-gram features of documents.

3.4.2 *Term Weighting*

As we have already presented, when the document is represented by the Bag-of-Words model, the term frequency (TF) criterion (or TF-IDF) constitutes the basis for weighting the terms of each document. How can this be done in the graph-based representation? The answer is given by utilizing node centrality criteria of the graph (Newman, 2010; Easley and Kleinberg, 2010). Given a graph representation, the importance of a term in a document can be inferred by the importance of the corresponding node in the graph.

In the previous section, we presented local and global centrality criteria that have been widely used for graph mining and network analysis purposes; here, we propose that those criteria can also be used for weighting terms in the TC task. That way, similar to TF, we can define the Term Weight (TW) weighting scheme as TW(t, d) = centrality(t, d), where centrality(t, d) corresponds to the score of term (node) t in the graph representation G_d of document d. The interesting point here is that TW can be used along with any centrality criterion in the graph, local or global.

Furthermore, we can extend this weighting scheme by considering information about the inverse document frequency (IDF factor) of the term t in the collection $\mathcal{D}$. That way, we can derive the TW-IDF model as follows:

$$\text{TW-IDF}(t, d) = \text{TW}(t, d) \times \text{IDF}(t, \mathcal{D}). \tag{3.2}$$

In fact, TW and TW-IDF constitute suites for graph-based term-weighting schemes and thus, can be applied in any text analytics task. Some of them have already been explored in graph-based IR (Rousseau and Vazirgiannis, 2013) and keyword extraction (Mihalcea and Tarau, 2004; Rousseau and Vazirgiannis, 2015).

A natural question here is what is the additive value of the TW and TW-IDF, compared to the widely used TF and TF-IDF. As we have already discussed, the graph-based representation and the corresponding weighting functions, question the term independence assumption that is imposed by the Bag-of-Words model and is inherited to the frequency-based schemes. The proposed weights are inferred from the interconnection of features (i.e., terms) – as suggested by the graph – and therefore information about n-grams is implicitly captured. That way, the feature space of the learning problem is kept to the one defined by the unique unigrams of our collection (instead of using simultaneously as features all the possible unigrams, bigrams, 3-grams, etc.), but the produced term weights incorporate n-gram information through the graph-based representation. In other words, the importance and discriminative power of a term is determined by a ranking process on the corresponding centrality metrics, in the interconnected feature space defined by the graph.

3.4.3 *Inverse Collection Weight (ICW)*

In the case of TF-IDF scheme, the frequency of each term in the document (TF factor) is penalized by the number of documents into which it appears (IDF factor). The same appearance-based IDF penalization mechanism can also be applied to the TW graph-based weight, and according to this the TW-IDF scheme of Eq. (3.2) has been derived. In this paragraph, we introduce the concept of Inverse Collection Weight (ICW) – a graph-based criterion to penalize the weight of terms that are "important" across the whole collection of documents. The main concept behind ICW is the collection-level graph $\mathcal{G}$ – an extension of the Graph-of-Words in the collection of documents $\mathcal{D}$.

Definition 3.1 (Collection Level Graph $\mathcal{G}$). *Let* $\{G_1, G_2, \ldots, G_d\}_{|\mathcal{D}|}$ *be the set of graphs that correspond to all documents* $d \in \mathcal{D}$*. The collection level graph* $\mathcal{G}$ *is defined as the union of graphs* $G_1 \cup G_2 \cup \ldots \cup G_d$ *over all documents in the collection.*

The union of two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ is defined as the union of their node and edge sets, i.e., $G \cup H = (V_G \cup V_H, E_G \cup E_H)$. The number of nodes in graph $\mathcal{G}$ is equal to the number of unique terms in the collection, while the number of edges is equal to the number of unique edges over all document-level graphs (see also Figure 3.2).

This graph captures the overall dependencies between the terms of the collection; the relative overall importance of a term in the collection will be proportional to the importance of the corresponding node in $\mathcal{G}$. Following similar methodological arguments as used for IDF (Robertson, 2004), we define a probability distribution over the nodes of $\mathcal{G}$ (or equivalently, the unique terms of $\mathcal{D}$), with respect to a centrality (term-weighting in our case) criterion; then, the probability of node (term) t will be:

$$\Pr(t) = \frac{\mathrm{TW}(t, \mathcal{D})}{\sum_{v \in \mathcal{D}} \mathrm{TW}(v, \mathcal{D})}. \tag{3.3}$$

Note that, in Eq. (3.3), we use $\mathcal{D}$ instead of $\mathcal{G}$; we consider that the space defined by the document collection $\mathcal{D}$ is equivalent to the one defined by graph $\mathcal{G}$ with respect to the unique terms of the collection. This way, the notion of $\mathrm{TW}(t, \mathcal{D})$ used here is consistent with what was described earlier. Based on this, we define the ICW measure as:

$$\mathrm{ICW}(t, \mathcal{D}) = \frac{\max_{v \in \mathcal{D}} \mathrm{TW}(v, \mathcal{D})}{\mathrm{TW}(t, \mathcal{D})}. \tag{3.4}$$

Instead of selecting the maximum centrality in the collection level (Eq. (3.4)), the sum of all centralities also yields good results.

ICW shares common intuition with the inverse total term frequency described in Robertson (2004). In fact, it can be considered as an extension of the total collection frequency of a term, to the graph-based document representation. Furthermore, similar to TW, it can be used along with any node centrality criterion.

Using ICW as a graph-based collection-level term penalisation factor, we derive a new class of term-to-document weighting mechanism, namely TW-ICW. This weighting scheme is derived combining different local (i.e., document-level) and global (i.e., collection-level) criteria as follows:

$$\text{TW-ICW}(t, d) = \text{TW}(t, d) \times \log(\text{ICW}(t, \mathcal{D})). \tag{3.5}$$

In the case of TW and ICW, any centrality criterion can be applied. However, the computational complexity is a crucial factor that should be taken into account. Nevertheless, as we have noticed from the experimental evaluation, even using simple and easy-to-compute local criteria (e.g., degree), we achieve good classification performance.

3.4.4 *Label Graphs*

Following previous work (Ko, 2012) on how to use class information for text classification, Shanavas et al. (2016) introduced supervised term weighting (TW-CRC) as a method to integrate class information with graphs. Similarly, we create a graph for each class (label), where we add all words of documents belonging to the respective class as nodes and their co-occurrence as edges. Our weighting scheme is a variant of TW-CRC; we define LW for a term t as:

$$\text{LW}(t) = \frac{\max(\deg(t, L))}{\max(\text{avg}(\deg(t, L)), \min(\deg(L))}, \tag{3.6}$$

where the maximum degree of term t in all label graphs (L) is divided by the max of two values: the average degree of the term in all label graphs (except the one having the max degree) and the min degree of all the terms in all the label graphs. Then, we obtain ICW-LW as follows:

$$\text{ICW-LW}(t, d) = \log(\text{ICW}(t, \mathcal{D}) \times \text{LW}(t)), \tag{3.7}$$

and multiply it with TW(t, d) to get TW-ICW-LW. Notice that, supervised frequency-based methods have also been proposed in previous work (Debole and Sebastiani, 2004; Huynh et al., 2011), where the Term Category Dependence Measure was introduced.

3.4.5 *Edge Weighting using Word Embeddings*

With our proposed framework, we can now use word embeddings (Bengio et al., 2003) in order to extract similarities between terms. Our goal is to integrate these similarities in the graph representation as weights on the edge between two words. The key idea behind our approach is that we want to reward semantically close words in the graph-document level (TW) and penalize them in the collection level (ICW).

The most commonly used similarity between two words t_1 and t_2 in the word-embedding space is cosine similarity, which ranges between -1 and 1. In order to have a valid distance metric, we need to bound this between 0 and 1. We use the angular similarity to represent the weight of an edge between two words, and since the vector elements may be positive or negative, the formula becomes:

$$\text{weight}(t_1, t_2) = 1 - \frac{\arccos(\text{sim}(t_1, t_2))}{\pi}. \tag{3.8}$$

A key problem that arises when using word embeddings, is how to deal with words that are not present in the vocabulary. `GloVe` (Pennington et al., 2014) includes an "unknown" word-vector for words that are not present in the word embedding space. In this case we will have a cosine similarity of one for two words that do not exist in the vocabulary. A solution, is finding synonyms of words using the `wordnet` tool.

The best performance was given by using Google's pre-trained word embeddings (Mikolov et al., 2013a) and not by learning them by the datasets. We remind that the learning is done on the training words. Since the words included in the pre-trained version of word2vec are case sensitive and not stemmed, we did not apply any of these transformations. For words that do not appear in word2vec, we add a small value as similarity. Other distances (e.g. inverse euclidean, fractional) did not yield any further improvement.

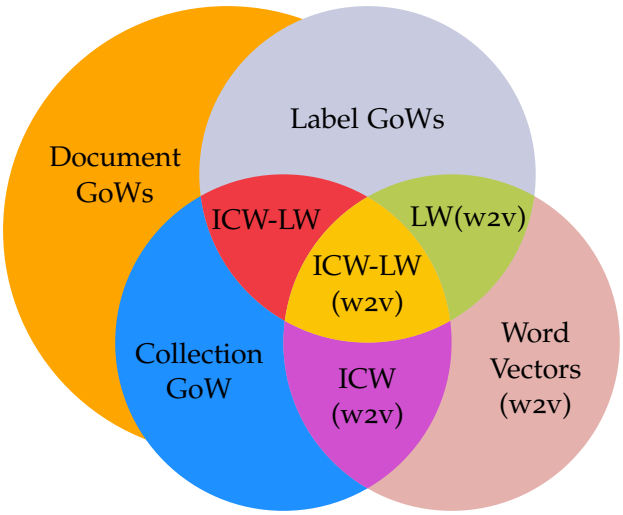


Figure 3.3: Blending different types of GoWs and word vector similarities in one framework. Our proposed metrics are located in the intersected areas.

A similar approach for generic keyphrase extraction can be tracked in work by Wang et al. (2015). Providing more information in the weights, like number of co-occurrences between words, did not yield better results.

3.4.6 *Computational complexity*

As described earlier, some formulations of the TW-based schemes consider centrality criteria that are computationally intensive (e.g., closeness centrality). Nevertheless, in the case of TW and TW-IDF, those criteria are applied on a per document basis, where the corresponding graphs are sparse and of very small size, and thus are not prohibitive (we have observed that, with very small window sizes ($w = \{2, 3\}$), we can achieve both high accuracy and low computation time). We stress out here that the local and computationally efficient degree centrality criterion – with complexity of same order as the one of term frequency – performs quite well in most of the cases. In any case, even the most intensive criteria considered here (such as the closeness centrality), can be efficiently approximated very well (Eppstein and Wang, 2004).

For the collection level graph $\mathcal{G}$ and the ICW-based schemes, the basic point is that the weighting of each term will be computed once during the training phase of

	Train	Test	Voc	Avg	#w2v	#ICW
IMDB	1,340	660	32,844	343	27,462	352K
WEBKB	2,803	1,396	23,206	179	20,990	273K
20NG	11,293	7,528	62,752	155	54,892	1.7M
AMAZON	5,359	2,640	19,980	65	19,646	274K
REUTERS	5,485	2,189	11,965	66	9,218	163K
SUBJ.	6,694	3,293	8,639	11	8,097	58K

Table 3.1: Datasets' statistics: #ICW shows the number of edges in the collection-level graph; #w2v: number of words that exist in pre-trained vectors.

the model; thus, in the testing phase, heavy computations are not performed. In a similar way as in the document-based (local) graphs, the overall execution time can be improved, relying on approximation techniques of the corresponding measures. Lastly, by applying the easy-to-compute feature selection method presented earlier, the feature space of the problem is reduced, improving the training time of the classifier.

Our implementation is also efficient for large datasets, as the graph construction and centrality computation processes, take advantage of multi-core machines and can be distributed evenly to all available threads.

An overview of our approach is shown in Figure 3.3.

3.5 EXPERIMENTS

In this section, we present the datasets and the experimental evaluation of the proposed TC framework. Here, we deal with the problem of multi-class, single-label text categorization, where each document is assigned exactly to one category.

We have evaluated our method on six freely available standard TC datasets, covering multi-class document categorization, sentiment analysis and subjectivity. Specifically:

1. 20NG[§]: newsgroup documents belonging to 20 categories,
2. REUTERS[§]: 8 categories of Reuters-21578,

§ web.ist.utl.pt/acardoso/datasets/

3. WEBKB[§]: 4 most frequent categories of webpages from Computer Science departments,

4. IMDB (Pang and Lee, 2004): positive and negative movie reviews;

5. AMAZON (Blitzer et al., 2007): product reviews acquired from Amazon over four different sub-collections;

6. SUBJECTIVITY (Pang and Lee, 2004): contains subjective sentences gathered from Rotten Tomatoes and objective sentences gathered from IMDB.

A summary of the datasets can be found in Table 3.1.

Since the goal of this work is to introduce new term weighting schemes, we rely on widely used classification algorithms. Specifically, we have used linear SVMs, due to their superior performance in TC (Joachims, 1998). Furthermore, as discussed in Leopold and Kindermann (2002), the choice of the kernel function of SVM is not very crucial, compared to the significance of the term weighting schemes. In the experiments, linear SVMs were used with grid search cross-validation for tuning the C parameter. We also examined logistic regression (LR), and observed similar performance. In the text preprocessing step, we have removed stopwords. No stemming or lowercase transformation was applied in order to match the most words in `word2vec`.

We compare the proposed weighting schemes to several baseline methods. (i) The TW weighting scheme to (a) binary n-gram features (denoted by TF binary), (b) the traditional TF weights (denoted by TF), (c) to centroid embeddings (w2v), and (d) TF-IDF aggregated with w2v features. (ii) The TW-IDF scheme to the well-known TF-IDF. (iii) The TW-ICW scheme with word embedding similarities as weights to TF-IDF combined with centroids of word embeddings.

We consider that comparing models of similar degree of complexity is more meaningful. Given the fact that TF and TF-IDF baselines perform well in general, we are also interested to have a more broad comparison of the performances, examining the best scheme for each dataset. For evaluation we use macro-average F1 score (F1) and classification accuracy (Acc) on the test sets; that way, we deal with the skewed class size distribution of some datasets (Sebastiani, 2002).

For the notation of the proposed schemes, we use TW (centrality measure) (e.g., TW (degree)) to indicate the centrality and TW-ICW (centrality at G, centrality

Methods	20NG (MAX)				IMDB (SUM)				SUBJECTIVITY (MAX)			
	$w = 3$		$w = 4$		$w = 2$		$w = 3$		$w = 6$		$w = 7$	
	F1	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**
TF	80.88	81.55	-		84.23	84.24	-		88.42	88.43	-	
w2v	74.43	75.75	-		82.57	82.57	-		87.67	87.67	-	
TF-binary (ngrams)	81.64	82.11*	-		83.02	83.03	-		87.51	87.51	-	
TW (degree)	82.37	83.00*	82.21	82.83*	84.82	84.84	84.67	84.69	88.33	88.33	89.00	89.00*
TW (w2v)	81.88	82.51*	82.21	82.87*	84.66	84.69	84.52	84.54	87.75	87.57	87.66	87.67
TF-IDF	82.44	83.01*	-		83.33	83.33	-		89.06	89.06*	-	
TF-IDF-w2v	82.52	83.09*	-		82.87	82.87	-		89.91	89.91*	-	
TW-IDF (degree)	84.75	85.47*	84.80	85.46*	82.86	82.87	83.02	83.03	89.33	89.34*	89.33	89.34*
TW-IDF (w2v)	84.66	85.32	84.46	85.13	83.47	83.48	83.31	83.33	86.42	86.42	86.51	86.51
TW-ICW (deg, deg)	85.24	85.80*	**85.41**	**86.05***	84.98	85.00	85.13	85.15	89.30	89.31*	89.61	89.61*
TW-ICW (w2v)	**85.33**	**85.93***	85.29	85.90*	85.12	85.15	84.82	84.84	89.61	89.61*	87.30	87.30
TW-ICW-LW (deg)	85.01	85.66*	85.02	85.66*	85.73	85.75	85.28	85.30	**90.12**	**90.13***	**90.27**	**90.28***
TW-ICW-LW (w2v)	82.56	83.11*	82.24	82.81*	85.29	85.30	84.39	84.39	87.70	87.70	87.70	87.70
TW-ICW-LW (pgr)	83.92	84.66	83.80	84.54	84.97	85.00	85.73	85.75	86.60	86.60	86.45	86.45
TW-ICW-LW (cl)	84.61	85.22	84.71	85.27	**87.27**	**87.27***	**86.06**	**86.06**	89.97	89.97*	90.09	90.10*

Table 3.2: Macro-F1 and accuracy for window size w. Bold shows the best performance on each window size and blue the best overall on each dataset. * indicates statistical significance of improvement over TF at $p < 0.05$ using micro sign test. MAX and SUM state the best numerator for ICW in Eq. (3.4).

at $\mathcal{G}$) (e.g., TW-ICW (degree, degree)) for the local and collection-level graphs respectively. In TW-IDF (w2v), we compute the weighted degree centrality on the document level, with word-embedding similarities as weights. Similarly, in TW-ICW (w2v) we compute both weighted centralities for document and collection graphs. Finally, we denote as TW-ICW-LW the blending of TW, ICW and label graphs (LW). In label graphs we only make use of the degree centrality, since it is fast and performs best.

RESULTS Tables 3.2 and 3.3 present the results concerning the categorization performance of the proposed schemes for the six datasets. As discussed previously, the size of the window considered to create the graphs is one of the model's parameters. From the extensive experimental evaluation that we have performed, we have concluded that small window sizes give the most persistent results across various datasets and weighting schemes. For completeness in the presentation, we report results for two window sizes. In order to capture more information, we need larger window sizes for small datasets (e.g. SUBJECTIVITY). Also, since for the baseline methods (TF, TF binary, TF-IDF, w2v, TF-IDF-w2v) there is no notion

Methods	Amazon (max)				WebKB (sum)				Reuters (max)			
	$w = 2$		$w = 3$		$w = 2$		$w = 3$		$w = 2$		$w = 3$	
	F1	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**
TF	80.68	80.68	-		90.31	91.91	-		91.51	96.34	-	
w2v	79.05	79.05	-		84.54	86.58	-		91.35	96.84	-	
TF-binary (ngrams)	79.84	79.84	-		91.22	92.85	-		86.33	95.34	-	
TW (degree)	80.07	80.07	80.41	80.41	91.69	92.64	91.45	92.49	93.58	97.53*	93.08	97.25*
TW (w2v)	80.07	80.07	79.54	79.54	91.70	92.64	91.00	92.06	93.09	97.35*	93.43	97.25*
TF-IDF	80.26	80.26	-		87.79	89.89	-		91.89	96.71	-	
TF-IDF-w2v	80.49	80.49	-		88.18	90.18	-		91.33	96.80	-	
TW-IDF (degree)	81.47	81.47*	81.55	81.55*	90.38	91.70	90.47	91.84	93.80	97.30*	93.13	97.35*
TW-IDF (w2v)	79.61	79.62	77.60	77.61	90.81	92.20	90.60	91.91	93.38	97.44*	**93.87**	**97.44***
TW-ICW (deg, deg)	82.08	82.08*	82.02	82.02*	91.72	92.78	91.42	92.49	92.91	97.35	93.59	97.39*
TW-ICW (w2v)	80.86	80.87*	78.82	78.82	91.58	92.64	91.84	92.85	93.57	97.30*	92.96	97.25
TW-ICW-LW (deg)	82.72	82.72*	82.91	82.91*	91.86	92.92	91.95	92.92	**93.88**	**97.53***	93.48	97.35*
TW-ICW-LW (w2v)	80.56	80.56	78.32	78.33	90.74	91.99	90.01	91.34	92.51	96.89	92.14	96.98
TW-ICW-LW (pgr)	82.23	82.23*	82.46	82.46*	91.18	92.20	92.23	93.07	93.38	97.35*	93.37	97.35*
TW-ICW-LW (cl)	**82.90**	**82.91***	**83.02**	**83.03***	**92.72**	**93.57***	**92.86**	**93.57***	93.12	97.25	92.87	97.21

Table 3.3: Macro-F1 and accuracy for different window size w. Bold shows the best performance for a sliding window and blue the best overall on each dataset.

of window size, the results for $w = \{2, 3\}$ are the same. We have also examined several centrality criteria (using both undirected and directed graphs); undirected giving better results.

Comparing TF to the graph-based ones, namely TW (degree), in almost all cases TW gives higher F1 and accuracy results. Similar observations can be made in the case where the IDF penalization is applied. In most of the datasets, the TW-IDF (degree) scheme performs quite well. The interesting point here, which is confirmed by the related literature (Lan et al., 2005), is that TF-IDF is in general inferior to TF in TC. However, when the IDF penalization factor is applied on the TW term-to-document weighting, a powerful mechanism is derived. In the case of purely graph-based schemes, we can observe that some of them produce very good classification results. In almost all cases, TW-ICW-LW (degree or closeness) achieve the best performance.

WORD2VEC SIMILARITIES Significant improvement is observed by adding the w2v similarities as weights in the document, collection level and label graphs in almost all datasets. In fact, we have obtained better results in 20NG (TW-ICW (w2v)), WebKB (TW-ICW (w2v)) and Reuters (TW-IDF(w2v)), by boosting semantically close words in the document level and penalizing them in the collection

	20NG	IMDB	SUBJ.	AMAZON	WEBKB	REUTERS
CNN (no w2v, 20 ep.) (Kim, 2014)	83.19	74.09	88.16	80.68	88.17	94.75
CNN (w2v, 20 ep.)	81.92	64.09	89.04	82.08	84.05	95.80
FastText (100 ep.) (Joulin et al., 2017)	79.70	84.70	88.60	79.50	92.60	97.00
FastText (w2v, 5 ep.)	80.80	86.10	88.50	80.90	91.40	97.40
TextRank (Mihalcea and Tarau, 2004)	82.56	83.33	84.78	80.49	92.27	97.35
Word Attraction (Wang et al., 2015)	61.24	70.75	86.60	78.29	79.46	91.34
TW-CRC (Shanavas et al., 2016)	85.35	85.15	89.28	81.13	92.71	97.39
TW-ICW-LW (**ours**)	**86.05**	**87.27**	**90.28**	**83.03**	**93.57**	**97.53**

Table 3.4: Comparison in accuracy(%) to deep learning and graph-based approaches.

level. It is evident that the graph-based criteria can further improve the classification task.

The discriminative nature of the features is derived by the underlying graph and by the fact that we treat the term weighting process as a ranking task in the interconnected feature space defined by the graph. We augment the unigram feature space of the learning task with weights that implicitly consider information of n-grams (short and long ones) in the document – as expressed by paths in the graph – without increasing the dimensionality of the problem. In other words, the feature space is the one defined by the unigrams of our collection, but the weights capture information beyond them.

TF n-gram binary scheme (TF binary) has also been examined, i.e., all the possible n-grams of the collection with binary values (up to 6-grams in our experiments). This has the effect that the feature space explodes (e.g. 20NG: 6M, REUTERS: 1.3M, WEBKB: 1.5M and IMDB: 3M) – with direct implication on the efficiency of the method.

For comparison reasons, the size of the unigram feature space considered by our framework is equal to the unique terms in the collections and much smaller compared to the $n-$grams ones. Moreover, graph-based weighting is able to outperform TF (binary) in all datasets.

We clearly see that by fusing document, collection and label graphs we obtain the best results in almost in 5 out of 6 datasets. Label graphs information consist a powerful weighting method, when combined with our proposed collection level graph approach.

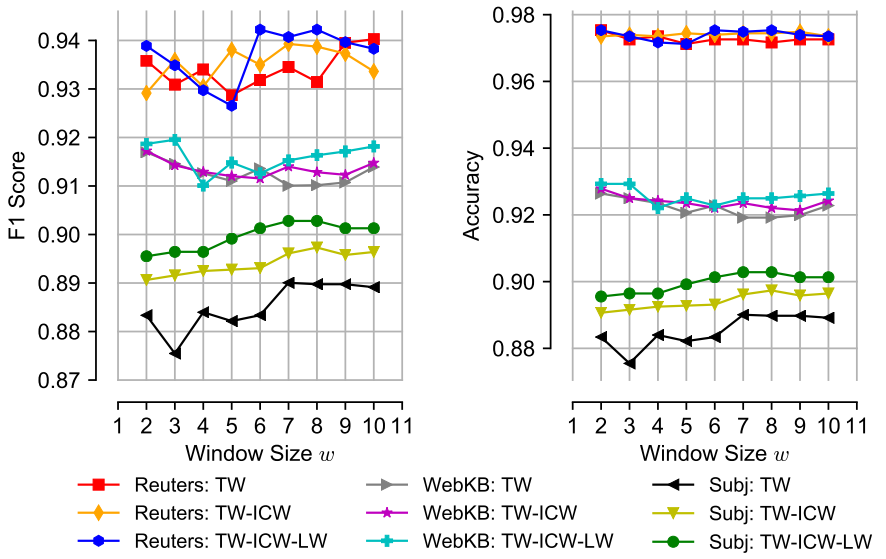


Figure 3.4: F1 score (left) and accuracy (right) of TW, TW-ICW and TW-ICW-LW (all degree) on REUTERS, WEBKB and SUBJECTIVITY, for window size $w = \{2, \ldots, 10\}$.

Adding word2vec similarities as weights, when label graphs are used, does not improve the accuracy. This implies that important terms concerning different labels can be close in the word vector space. Choosing closeness in the document level GoW yields the best performance in 3 datasets. Closeness can only have an affect in larger document lengths and when used along with label graphs.

COMPARISON To further investigate the effectiveness of our approach, we have compared our results with current state-of-the-art graph-based and non graph-based methods. In Table 3.4 we compare against CNN for text classification, without pre-trained word vectors (Kim, 2014), FastText (Joulin et al., 2017), TextRank (Mihalcea and Tarau, 2004), Word Attraction weights based on word2vec similarities (Wang et al., 2015) and Supervised Term Weighting (TW-CRC) by Shanavas et al. (2016). Our work produces better or comparable results to their work. Since the implementation of most models is our own, their performance is not optimal.

SLIDING WINDOW IN GOW Selecting the window size w is also important. As we observed, the maximum accuracy is achieved while using small window sizes.

autos	TF-IDF	pos: Jeep, Opel, eliot, clas, Intrepid neg: bike, edu, University, College, David
	TW-ICW	pos: driving, autos, Ford, Toyota, automotive neg: bike, DoD, bikes, guns, parking
electr.	TF-IDF	pos: circuits, phone, Yingbin, Radar, Shack neg: Windows, university, com, forsale, edu
	TW-ICW	pos: power, circuits, detector, voltage, scope neg: forsale, unused, modem, sale, shipping

Table 3.5: Largest weights in TF-IDF and TW-ICW for subsets of 20NG.

In any case, even if larger values of w were able to get slightly better results, a smaller window size would be preferable, due to the overall overhead that could be introduced (increase of the density of the graph). Figure 3.4 depicts the F1 score and accuracy on the WEBKB, REUTERS and SUBJECTIVITY datasets, using the TW, TW-ICW and TW-ICW-LW(deg) schemes for various window sizes. We notice also that larger sliding windows are only improving accuracy in datasets with small document length (e.g. SUBJECTIVITY).

EXPRESSIVENESS In Table 3.5 we present the largest weights connected with their respective words for two subsets of 20NG. One can easily observe that TW-ICW manages to associate bigger weights with more expressive words than TF-IDF. TW-ICW succeeds to capture more meaningful words as more important for both positive and negative class, compared to TF-IDF, which outputs similar words for the negative class across different subsets.

3.6 CONCLUSION & FUTURE WORK

In this chapter, we proposed a full graph-based framework for text classification. By treating the term weighting task as a node ranking problem of interconnected features defined by a graph, we were able to determine the importance of a term using node centrality criteria. Building on this formulation, we introduced simple-yet-effective weighting schemes at the collection and label level, in order to penalize globally important terms (as analogous to “globally frequent terms") and reward locally important terms respectively. We also incorporate additional word-embedding information as weights in the graph-based representations.

Our proposed methods could also be applied in information retrieval, as well as other domains. In fact, document-level graph-based term weighting has already been applied there, so it would be interesting to examine the performance of the proposed collection-level (ICW) penalization mechanism. In the unsupervised scenario, where label information is not available, community detection algorithms may be applied to identify clusters of words or documents in collection graphs. Graph-based representations of text could also be fitted into deep learning architectures following the idea of Lei et al. (2015). Alternatively, they can be passed to neural message passing architectures (Gilmer et al., 2017). Lastly, one could examine a Graph-of-Documents approach, in which we create a graph, where nodes represent documents and edges correspond to similarity between them. In this case, graph kernels could be utilized for graph comparison and/or Word Mover's distance (Kusner et al., 2015) between two documents as weights.

4

REGULARIZATION FOR TEXT CLASSIFICATION

Regularization is a critical step in supervised learning to not only address overfitting, but also to take into account any prior knowledge we may have on the features and their dependence. In this chapter, we explore diverse linguistic groups, like clusters in the word embedding space, for structured regularization, as well as new regularizers for standard classifiers, which can come from text representations studied previously. We show that our proposed methods are comparable to state-of-the-art ones in terms of accuracy while producing sparser models.

4.1 INTRODUCTION

Harnessing the full potential in text data has always been a key task for the natural language processing and machine learning communities. The properties hidden under the inherent high dimensionality of text are of major importance in tasks such as text categorization and opinion mining.

Although simple models like Bag-of-Words (BoW) manage to perform well, the problem of overfitting still remains. Regularization, as proven in Chen and Rosenfeld (2000), is of paramount importance in Natural Language Processing and more specifically language modeling, structured prediction, and classification. In this work we build upon the paper of Yogatama and Smith (2014b) who introduce prior knowledge of data as a regularization term. One of the most popular structured regularizers, the group lasso (Yuan and Lin, 2006), was proposed to avoid large L2 norms for groups of weights.

Our contribution will consist of novel linguistic structured regularizers that capitalize on the clusters learned from texts using the word2vec and Graph-of-Words (GoW) document representation, which can be seen as group lasso variants, as well as new algorithms for structured regularization. The extensive experiments we conducted demonstrate that these regularizers, along with the our new proposed regularization algorigthms, can boost standard Bag-of-Words models on

most cases tested in the task of text categorization, by imposing additional unused information as bias.

4.2 RELATED WORK

In this part, we review particularly relevant prior work on regularization for text classification and more specifically methods based on group lasso.

In many applications of statistics and machine learning, the number of exploratory variables may be very large, while only a small subset may truly be relevant in explaining the response to be modelled. In certain cases, the dimensionality of the predictor space may also exceed the number of examples. Then the only way to avoid overfitting is via some form of "capacity control" over the family of dependencies being explored. Estimation of sparse models that are supported on a small set of input variables is thus highly desirable, with the additional benefit of leading to parsimonious models, which can be used not only for predictive purposes but also to understand the effects (or lack thereof) of the candidate predictors on the response.

More specifically, regularization in text scenarios is essential as it can lead to removing unnecessary words along with their weights. For example, in text classification, we may only care for a small subset of the vocabulary that is important during the learning process, by penalizing independently or in grouped way noisy and irrelevant words.

With noiseness we refer to words that may increase the dimensionality and complexity of a problem, while having a clear decreasing effect in performance.

Another example task is text normalization, where we want to transform lexical variants of words to their canonical forms. Text normalization can be seen as a machine learning problem (Ikeda et al., 2016) and thus regularization techniques can be applied.

Next we present standard regularization methods, which prove to be effective for classification tasks. We also use them later as baselines for our experiments.

L_1 AND L2 REGULARIZATION The two most used penalty terms are known as L_1 regularization, a. k. a. *lasso* (Tibshirani, 1996), and L2 regularization, a. k. a.

ridge (Hoerl and Kennard, 1970) as they correspond to penalizing the model with respectively the L_1 and L2 norm of the feature weight vector θ:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(x^i, \theta, y^i) + \lambda \sum_{j=1}^{p} |\theta_j| \tag{4.1}$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(x^i, \theta, y^i) + \lambda \sum_{j=1}^{p} \theta_j{}^2 \tag{4.2}$$

ELASTIC NET A linear combination of the L_1 and L2 penalties has been also introduced by Zou and Hastie (2005), called *elastic net*. Although L_1 and elastic net can be very effective in terms of sparsity, the accuracy achieved by these regularizers can be low. On the contrary, L2 can deliver sufficient accuracy at the cost of zero sparsity. The need for new methods that outperform the aforementioned approaches in both accuracy and sparsity is evident.

GROUP STRUCTURED REGULARIZATION In many problems a predefined grouping structure exists within the explanatory variables, and it is natural to incorporate the prior knowledge so that the support of the model should be a union over some subset of these variable groups. Group structured regularization has been proposed to address the problem of overfitting, given we are provided with groups of features. Group lasso is a special case of group regularization proposed by Yuan and Lin (2006), to avoid large L2 norms for groups of weights, given we are provided with groups of features. The main idea is to penalize together features that may share some properties.

Group structured regularization or variable group selection problem is a well-studied problem, based on minimizing a loss function penalized by a regularization term designed to encourage sparsity at the variable group level. Specifically, a number of variants of the L_1-regularized lasso algorithm (Tibshirani, 1996) have been proposed for the variable group selection problem, and their properties have been extensively studied recently. First, for linear regression, Yuan and Lin (2006) proposed the group lasso algorithm as an extension of lasso, which minimizes the squared error penalized by the sum of L2-norms of the group variable coefficients across groups. Here the use of L2-norm within the groups and L_1-norm across the groups encourages sparsity at the group level.

In addition, group lasso has been extended to logistic regression for binary classification, by replacing the squared error with the logistic error (Kim et al., 2006b; Meier et al., 2008), and several extensions thereof have been proposed (Roth and Fischer, 2008).

Later, sparse group lasso and overlapping group lasso were introduced (Jacob et al., 2009; Obozinski et al., 2011) to additionally penalize features inside the groups, while the latter can be used when groups include features that can be shared between them.

LINGUISTIC STRUCTURED REGULARIZERS As mentioned previously, words that appear together in the same context, share topics or even have a similar meaning, may form groups that capture semantic or syntactic prior information. Hence we can feed these groups to group lasso. Yogatama and Smith (2014a) used the Alternating Direction Method of Multipliers algorithm (ADMM) (Boyd et al., 2011) for group lasso, an algorithm that solves convex optimization problems by breaking them into smaller pieces. In their paper, groups extracted by Latent Dirichlet Allocation (LDA) and sentences were used for structured regularization.

While current state-of-the-art methods either focus on finding the most meaningful groups of features or how to further "optimize" the group lasso approach, the attempts carry as well the disadvantages of group lasso architectures. In some cases, we may not be able to extract "good" groups of words. As presented in the next section, we want to explore new ways of regularization on groups, diverging from group lasso, that can give high accuracy with high sparsity.

4.3 BACKGROUND

We place ourselves in the scenario where we consider a prediction problem, in our case text categorization, as a loss minimization problem, i. e., we define a loss function $\mathcal{L}(x, \theta, y)$ that quantifies the loss between the prediction $h_\theta(x)$ of a classifier parametrized by a vector of feature weights θ and a bias b, and the true class label $y \in \mathcal{Y}$ associated with the example $x \in \mathcal{X}$. Given a training set of N

data points $\{(x^i, y^i)\}_{i=1...N}$, we want to find the optimal set of feature weights θ^* such that:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{N} L(x^i, \theta, y^i)}_{\textbf{empirical risk}} \tag{4.3}$$

In the case of logistic regression with binary predictions ($\mathcal{Y} = \{-1, +1\}$), $h_\theta(x) = \theta^\top x + b$ and $\mathcal{L}(x, \theta, y) = \log[1 + e^{-y h_\theta(x)}]$ (log loss).

4.3.1 *Regularization*

Only minimizing the empirical risk can lead to overfitting, that is, the model no longer learns the underlying pattern we are trying to capture but fits the noise contained in the training data and thus results in poorer generalization (e. g., lower performances on the test set). For instance, along with some feature space transformations to obtain non-linear decision boundaries in the original feature space, one could imagine a decision boundary that follows every quirk of the training data. Additionally, if two hypothesis lead to similar low empirical risks, one should select the "simpler" model for better generalization power, simplicity assessed using some measure of model complexity.

LOSS+PENALTY Regularization takes the form of additional constraints to the minimization problem, i. e., a budget on the feature weights, which are often relaxed into a penalty term $\Omega(\theta)$ controlled via a Lagrange multiplier λ. We refer to the book of Boyd and Vandenberghe (2004) for the theory behind convex optimization. Therefore, the overall expected risk (Vapnik, 1991) is the weighted sum of two components: the empirical risk and a regularization penalty term, expression referred to as "Loss+Penalty" by Hastie et al. (2009). Given a training set of N data points $\{(x^i, y^i)\}_{i=1...N}$, we now want to find the optimal set of feature weights θ^* such that:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underbrace{\underbrace{\sum_{i=1}^{N} \mathcal{L}(x^i, \theta, y^i)}_{\textbf{empirical risk}} + \underbrace{\lambda\Omega(\theta)}_{\textbf{penalty term}}}_{\textbf{expected risk}} \tag{4.4}$$

L_1 AND L2 REGULARIZATION As also mentioned previously, two of the most used regularization schemes are L_1 regularization, called *Lasso* (Tibshirani, 1996) or *basis pursuit* in signal processing (Chen et al., 2001), and L2 regularization, called *ridge* (Hoerl and Kennard, 1970) or *Tikhonov* (Tikhonov and Arsenin, 1977).

PRIOR ON THE FEATURE WEIGHTS L_1 (resp. L2) regularization can be interpreted as adding a Laplacian (resp. Gaussian) prior on the feature weight vector. Indeed, given the training set, we want to find the most likely hypothesis $h^* \in \mathcal{H}$, i. e., the one with *maximum a posteriori* probability:

$$
\begin{aligned}
h^* &= \arg\max_{h \in \mathcal{H}} \left(\mathbb{P}(h | \{(x^i, y^i)\}_{i=1\dots N}) \right) \\
&= \arg\max_{h \in \mathcal{H}} \left(\frac{\mathbb{P}(\{y^i\}_i | \{x^i\}_i, h)\, \mathbb{P}(h | \{x^i\}_i)}{\mathbb{P}(\{y^i\}_i | \{x^i\}_i)} \right) \\
&= \arg\max_{h \in \mathcal{H}} \left(\mathbb{P}(\{y^i\}_i | \{x^i\}_i, h)\, \mathbb{P}(h | \{x^i\}_i) \right) \\
&= \arg\max_{h \in \mathcal{H}} \left(\mathbb{P}(\{y^i\}_i | \{x^i\}_i, h)\, \mathbb{P}(h) \right) && (4.5) \\
&= \arg\max_{h \in \mathcal{H}} \left(\prod_{i=1}^{N} \left(\mathbb{P}(y^i | x^i, h) \right) \mathbb{P}(h) \right) && (4.6) \\
&= \arg\max_{h \in \mathcal{H}} \left(\sum_{i=1}^{N} \left(\log \mathbb{P}(y^i | x^i, h) \right) + \log \mathbb{P}(h) \right) \\
&= \arg\min_{h \in \mathcal{H}} \left(\underbrace{\sum_{i=1}^{N} \left(-\log \mathbb{P}(y^i | x^i, h) \right)}_{\textbf{empirical risk}} \underbrace{- \log \mathbb{P}(h)}_{\textbf{penalty term}} \right)
\end{aligned}
$$

For the derivation, we assumed that the hypothesis h does not depend on the examples alone (Eq. 4.5) and that the N training labeled examples are drawn from an i.i.d. sample (Eq. 4.6). In that last form, we see that the loss function can be interpreted as a negative log-likelihood and the regularization penalty term as a negative log-prior over the hypothesis. Therefore, if we assume a multivariate Gaussian prior on the feature weight vector of mean vector $\mathbf{0}$ and covariance matrix

$\Sigma = \sigma^2 I$ (i. e., independent features of same prior standard deviation σ), we do obtain the L2 regularization:

$$\mathbb{P}(h) = \frac{1}{\sqrt{(2\pi)^p |\Sigma|}} e^{-\frac{1}{2}\theta^\top \Sigma^{-1}\theta} \tag{4.7}$$

$$\Rightarrow -\log \mathbb{P}(h) = \frac{1}{2\sigma^2}\theta^\top I\theta + \frac{p}{2}\log(2\pi\sigma)$$

$$\overset{\text{argmax}}{=} \lambda \|\theta\|_2{}^2, \quad \lambda = \frac{1}{2\sigma^2} \tag{4.8}$$

And similarly, if we assume a multivariate Laplacian prior on the feature weight vector (i. e., $\theta_i \sim Laplace(0, \frac{1}{\lambda})$), we obtain L_1-regularization. In practice, in both cases, the priors basically mean that we expect weights around 0 on average. The main difference between L_1 and L2 regularization is that the Laplacian prior will result in explicitly setting some feature weights to 0 (feature sparsity) while the Gaussian prior will only result in reducing their values (shrinkage).

4.3.2 *Structured Regularization*

In L_1 and L2 regularizations, features are considered as independent, which makes sense without any additional prior knowledge. However, similar features have similar weights in the case of linear classifiers – equal weights for redundant features in the extreme case – and therefore, if we have some prior knowledge on the relationships between features, we should include that information for better generalization, i. e., include it in the regularization penalty term.

Depending on how the similarity between features is encoded, e. g., through sets, trees (Kim and Xing, 2010; Liu and Ye, 2010; Mairal et al., 2010) or graphs (Jenatton et al., 2010), the penalization term varies but in any case, we take into account the structure between features, hence the "structured regularization" terminology. It should not be confused with "structured prediction" where this time the outcome is a structured object as opposed to a scalar (e. g., a class label) classically.

GROUP LASSO Bakin (1999) and later Yuan and Lin (2006) proposed an extension of L_1 regularization to encourage groups of features to either go to zero (as a

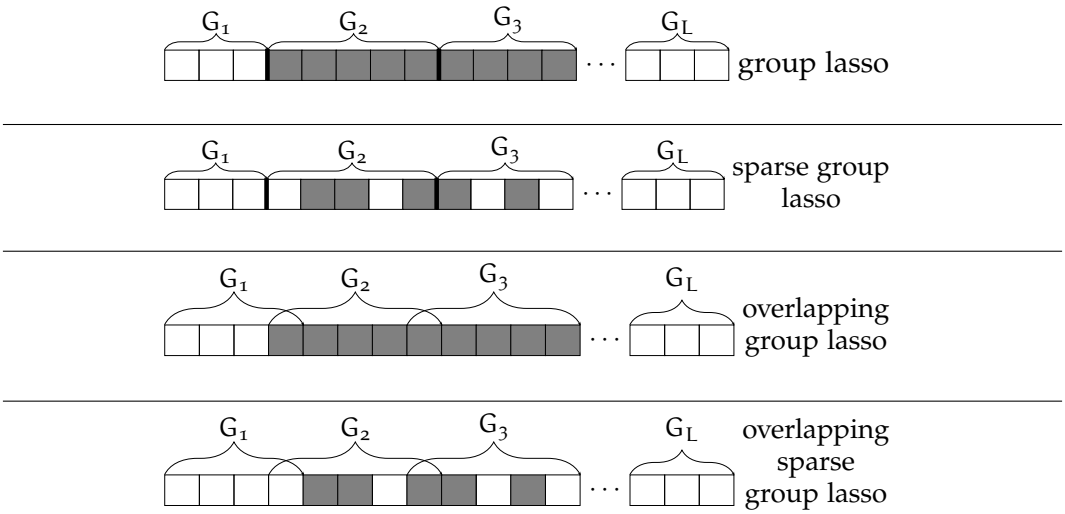


Figure 4.1: A graphical representation of different group lasso architectures. Grey boxes depict active features. While group lasso selects a whole group, sparse group lasso can select part of the group's features. In the overlapping case, groups can share features, while in the last case L_1 is applied inside each group.

group) or not (as a group), introducing *group* sparsity in the model. To do so, they proposed to regularize with the $L_{1,2}$ norm of the feature weight vector:

$$\Omega(\theta) = \lambda \sum_g \lambda_g \|\theta_g\|_2 \tag{4.9}$$

where θ_g is the subset of feature weights restricted to group g. Note that the groups can be overlapping (Jacob et al., 2009; Schmidt and Murphy, 2010; Jenatton et al., 2011; Yuan et al., 2011) even though it makes the optimization harder.

In Figure 4.1, we illustrate the selection of features by the most used group lasso regularizers. In group lasso, a group of features is selected and all its features are used. Next, in the sparse group lasso case, groups of features are selected again but not all the features belonging to them are used. In the overlapping group lasso, groups can share features between them. Finally, we may have sparse group lasso with overlaps.

Algorithm 1 ADMM for overlapping group-lasso

Input: augmented Lagrangian variable ρ, regularization strengths λ_{glas} and λ_{las}

while update in weights not small **do**
  $\theta = \underset{\theta}{\operatorname{argmin}}\, \Omega_{las}(\theta) + \mathcal{L}(\theta) + \frac{\rho}{2} \sum_{i=1}^{V} N_i(\theta_i - \mu_i)^2$
  for $g = 1$ to G **do**
    $v_g = \text{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(z_g)$
  end for
  $u = u + \rho(v - M\theta)$
end while

4.3.3 *Learning*

In our case we use a logistic regression loss function in order to integrate our regularization terms easily.

$$\mathcal{L}(x, \theta, y) = \log(1 + \exp(-y\theta^T x)) \tag{4.10}$$

It is obvious that the framework can be extended to other loss functions (e. g., hinge loss).

For the case of structured regularizers, there exist a plethora of optimization methods, such as group lasso. Since our tasks involve overlapping groups, we select the method of Yogatama and Smith (2014b). Their method uses the alternating directions method of multipliers (Hestenes, 1969; Powell, 1969).

Now given the lasso penalty for each feature and the group lasso regularizer, we get:

$$\min_{\theta, v} \Omega_{las}(\theta) + \Omega_{glas}(v) + \sum_{d=1}^{D} \mathcal{L}(x_d, \theta, y_d) \tag{4.11}$$

so that $\mathbf{v} = M\theta$, where $\mathbf{v}$ is a copy-vector of θ. The copy-vector $\mathbf{v}$ is needed because the group-lasso regularizer contains overlaps between the used groups. M is an indicator matrix of size $L \times V$, where L is the sum of the total sizes of all groups, and its ones show the link between the actual weights θ and their copies

$\mathbf{v}$. Following Yogatama and Smith (2014b), a constrained optimization problem is formed, that can be transformed to an augmented Lagrangian problem:

$$\Omega_{las}(\theta) + \Omega_{glas}(v) + \mathcal{L}(\theta) + u^\top(v - M\theta) + \frac{\rho}{2}\|v - M\theta\|_2^2 \tag{4.12}$$

Essentially, the problem becomes the iterative update of θ, v and u:

$$\min_\theta \Omega_{las}(\theta) + \mathcal{L}(\theta) + u^\top M\theta + \frac{\rho}{2}\|v - M\theta\|_2^2 \tag{4.13}$$

$$\min_v \Omega_{glas}(v) + u^\top v + \frac{\rho}{2}\|v - M\theta\|_2^2 \tag{4.14}$$

$$u = u + \rho(v - M\theta) \tag{4.15}$$

A schematic pseudocode presenting ADMM for sparse overlapping group is shown in Algorithm 1.

CONVERGENCE Yogatama and Smith (2014b) proved that ADMM for sparse overlapping group lasso converges. It is also shown that a good approximate solution is reached in a few tens of iterations. Our experiments confirm this as well.

4.4 REGULARIZING TEXT CLASSIFICATION WITH CLUSTERS OF WORDS

In this section we explore state-of-the-art structured regularizers and propose novel ones based on clusters of words from Latent Semantic Indexing topics, word2vec embeddings and Graph-of-Words document representation. We show that our proposed regularizers are faster than the state-of-the-art ones and still improve text classification accuracy. All code and data are available online*.

4.4.1 *Proposed framework*

In recent efforts there are results to identify useful structures in text that can be used to enhance the effectiveness of the text categorization in a NLP context. Since the main regularization approach we are going to use are variants of the group

* https://goo.gl/mKqvro

lasso, we are interested on prior knowledge in terms of groups/clusters that can be found in the training text data. These groups could capture either semantic, or syntactic structures that affiliate words to communities. In our work, we study both semantic and syntactic properties of text data, and incorporate them in structured regularizer. The grouping of terms is produced by either LSI or clustering in the word2vec or Graph-of-Words space.

4.4.1.1 *Statistical regularizers*

In this section, we present *statistical* regularizers, i. e., with groups of words based on co-occurrences, as opposed to *syntactic* ones (Mitra et al., 1997).

NETWORK OF FEATURES Sandler et al. (2009) introduced regularized learning with networks of features. They define a graph G whose edges are non-negative with larger weights indicating greater similarity. Conversely, a weight of zero means that two features are not believed a priori to be similar. Previous work (Ando and Zhang, 2005; Raina et al., 2006; Krupka and Tishby, 2007) shows such similarities can be inferred from prior domain knowledge and statistics computed on unlabeled data.

The weights of G are mapped in a matrix P, where $P_{ij} \geqslant 0$ gives the weight of the directed edge from vertex i to vertex j. The out-degree of each vertex is constrained to sum to one, $\sum_j P_{ij} = 1$, so that no feature "dominates" the graph.

$$\Omega_{network}(\theta) = \lambda_{net} \sum \theta_k^\top M \theta_k \tag{4.16}$$

where $M = \alpha(I - P)^\top (I - P) + \beta I$. The matrix M is symmetric positive definite, and therefore it possesses a Bayesian interpretation in which the weight vector θ, is *a priori* normally distributed with mean zero and covariance matrix $2M^{-1}$. However, preliminary results show poorer performance compared to structured regularizers in larger datasets.

SENTENCE REGULARIZER Yogatama and Smith (2014b) proposed to define groups as the sentences in the training dataset. The main idea is to define a group $d_{d,s}$ for every sentence s in every training document d so that each group holds

weights for occurring words in its sentence. Thus a word can be a member of one group for every distinct (training) sentence it occurs in. The regularizer is:

$$\Omega_{sen}(\theta) = \sum_{d=1}^{D} \sum_{s=1}^{S_d} \lambda_{d,s} \|\theta_{d,s}\|_2 \tag{4.17}$$

where S_d is the number of sentences in document d.

Since modern text datasets typically contain thousands of sentences and many words appear in more than one sentence, the sentence regularizer could potentially lead to thousands heavily overlapping groups. As stated in the work of Yogatama and Smith (2014b), a rather important fact is that the regularizer will force all the weights of a sentence, if it is recognized as irrelevant. Respectively, it will keep all the weights of a relevant sentence, even though the group contains unimportant words. Fortunately, the problem can be resolved by adding a lasso regularization (Friedman et al., 2010).

4.4.1.2 *Semantic regularizers*

In this section, we present *semantic* regularizers that define groups based on how semantically close words are.

LDA REGULARIZER Yogatama and Smith (2014a) considered topics as another type of structure. It is obvious that textual data can contain a huge number of topics and especially topics that overlap each other. Again the main idea is to penalize weights for words that co-occur in the same topic, instead of treating the weight of each word separately.

Having a training corpus, topics can be easily extracted with the help of the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003). In our experiments, we form a group by extracting the n most probable words in a topic. We note that the extracted topics can vary depending the text preprocessing methods we apply on the data.

LSI REGULARIZER Latent Semantic Indexing (LSI) can also be used in order to identify topics or groups and thus discover correlation between terms (Deerwester et al., 1990). LSI uses singular value decomposition (SVD) on the document-term matrix to identify latent variables that link co-occurring terms with documents.

The main basis behind LSI is that words being used in the same contexts (i. e., the documents) tend to have similar meanings. We used LSI as a baseline and compare it with other standard baselines as well as other proposed structured regularizers. In our work, we keep the top 10 words which contribute the most in a topic.

The regularizer for both LDA and LSI is:

$$\Omega_{\text{LDA, LSI}}(\theta) = \sum_{k=1}^{K} \lambda \|\theta_k\|_2 \tag{4.18}$$

where K is the number of topics.

4.4.1.3 *Graphical regularizers*

In this section we present our proposed regularizers based on Graph-of-Words and word2vec. Essentially the word2vec space can be seen as a large graph where nodes represent terms and edges similarities between them.

GRAPH-OF-WORDS REGULARIZER Following the idea of the network of features, we introduce a simpler and faster technique to identify relationships between features. We create a big collection graph from the training documents, where the nodes correspond to terms and edges correspond to co-occurrence of terms in a sliding window. The idea comes from the work (Skianis et al., 2018a) presented in the previous Chapter, where the collection-level GoW was introduced. An example of a Graph-of-Words representation lies in Figure 3.1.

A critical advantage of Graph-of-Words is that it easily encodes term dependency and term order (via edge direction). The strength of the dependence between two words can also be captured by assigning a weight to the edge that links them.

Graph-of-Words was originally an idea of Mihalcea and Tarau (2004) and Erkan and Radev (2004) who applied it to the tasks of unsupervised keyword extraction and extractive single document summarization. Rousseau and Vazirgiannis (2013) and Malliaros and Skianis (2015) showed it performs well in the tasks of information retrieval and text categorization. Notably, the former effort ranked nodes based on a modified version of the PageRank algorithm.

COMMUNITY DETECTION ON GRAPH-OF-WORDS Our goal is to identify groups or communities of words. Having constructed the collection-level Graph-of-Words, we can now apply community detection algorithms (Fortunato, 2010).

In our case we use the Louvain method, a community detection algorithm for non-overlapping groups described in the work of Blondel et al. (2008). Essentially it is a fast modularity maximization approach, which iteratively optimizes local communities until we reach optimal global modularity given some perturbations to the current community state. The regularizer becomes:

$$\Omega_{gow}(\theta) = \sum_{c=1}^{C} \lambda \|\theta_c\|_2 \tag{4.19}$$

where c ranges over the C communities. Thus θ_c corresponds to the sub-vector of θ such that the corresponding features are present in the community c. Note that in this case we do not have overlapping groups, since we use a non-overlapping version of the algorithm.

As we observe that the collection-level Graph-of-Words does not create well separated communities of terms, overlapping community detection algorithms, like the work of Xie et al. (2013) fail to identify “good" groups and do not offer better results.

WORD2VEC REGULARIZER Mikolov et al. (2013b) proposed the word2vec method for learning continuous vector representations of words from large text datasets. Word2vec manages to capture morpho-syntactic and semantic aspects of words and map them to a multi-dimensional vector space, giving the possibility of applying vector operations on them. We introduce another novel regularizer method, by applying unsupervised clustering algorithms on the word2vec space.

CLUSTERING ON WORD2VEC Pre-trained word embedding models contain millions of words represented as vectors. Since word2vec succeeds in capturing semantic and morpho-syntactic similarity between words, semantically related words tend to group together and create large clusters that can be interpreted as “topics".

In order to extract these groups, we use a fast clustering algorithm such as K-Means (MacQueen, 1967) and especially Minibatch K-means. The regularizer is:

$$\Omega_{word2vec}(\theta) = \sum_{k=1}^{K} \lambda \|\theta_k\|_2 \tag{4.20}$$

where K is the number of clusters we extracted from the word2vec space.

Clustering these semantic vectors is a very interesting area to study and could be a research topic by itself. The actual clustering output could vary as we change the number of clusters we are trying to identify. In this work we do not focus on optimizing the clustering process, as the number of clustering algorithms and parameters is huge.

4.4.2 *Experiments*

We evaluated our structured regularizers on several well-known datasets for the text categorization task. Table 4.1 summarizes statistics about the ten datasets we used in our experiments.

DATASETS From the 20 Newsgroups[†] dataset, we examine four binary classification tasks. We end up with binary classification problems, where we classify a document according to two related categories: comp.sys: ibm.pc.hardware vs. mac.hardware; rec.sport: baseball vs. hockey; sci: med vs. space and alt.atheism vs. soc.religion.christian. We use the 20NG dataset from the scikit-learn[‡] library in Python.

SENTIMENT ANALYSIS. The sentiment analysis datasets we examined include movie reviews (Pang and Lee, 2004; Zaidan and Eisner, 2008)[§], floor speeches by U.S. Congressmen deciding "yea"/"nay" votes on the bill under discussion (Thomas et al., 2006)[§] and product reviews from Amazon (Blitzer et al., 2007)[¶].

† http://qwone.com/~jason/20Newsgroups/
‡ https://scikit-learn.org/
§ http://www.cs.cornell.edu/~ainur/data.html
¶ http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

	Dataset	Train	Dev	Test	# Words	# Sents
20NG	science	949	238	790	25787	16411
	sports	957	240	796	21938	14997
	religion	863	216	717	18822	18853
	comp.	934	234	777	16282	10772
Sentiment	vote	1175	257	860	19813	43563
	movie	1600	200	200	43800	49433
	books	1440	360	200	21545	13806
	dvd	1440	360	200	21086	13794
	electr.	1440	360	200	10961	10227
	kitch.	1440	360	200	9248	8998

Table 4.1: Descriptive statistics of the datasets.

EXPERIMENTAL SETUP As features we use unigram frequency concatenated with an additional unregularized bias term.

Logistic regression was only considered, because of its interpretation for L2 regularizers as Gaussian prior on the feature weights and following Sandler et al. (2009), we considered a non-diagonal covariance matrix for L2 based on word similarity before moving to group lasso as presented in the paper. We are not expecting a significant change in results with different loss functions as the proposed regularizers are not log loss specific.

We reproduce standard regularizers like lasso, ridge, elastic and state-of-the-art structured regularizers like sentence, LDA as baselines and compare them with our proposed methods.

To extract LSI, LDA and word2vec groups we use the Gensim package (Řehůřek and Sojka, 2010) in Python. For the learning part we used Matlab and specifically code by Schmidt et al. (2007).

We split the training set in a stratified manner to retain the percentage of classes. We use 80% of the data for training and 20% for validation.

All the hyperparameters are tuned on the development dataset, using accuracy as the evaluation criterion. For lasso and ridge regularization, we choose λ from $\{10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$. For elastic net, we perform grid search on the same set of values as ridge and lasso experiments for λ_{ridge} and λ_{lasso}.

	dataset	no reg.	lasso	ridge	elastic	group lasso				
						LDA	LSI	sentence	GoW	word2vec
20NG	science	0.946	0.916	0.954	0.954	**0.968**	**0.968***	0.942	0.967*	**0.968***
	sports	0.908	0.907	0.925	0.920	0.959	0.964*	**0.966**	0.959*	0.946*
	religion	0.894	0.876	0.895	0.890	0.918	0.907*	**0.934**	0.911*	0.916*
	computer	0.846	0.843	0.869	0.856	0.891	0.885*	0.904	0.885*	**0.911***
Sentiment	vote	0.606	0.643	0.616	0.622	**0.658**	0.653	0.656	0.640	0.651
	movie	0.865	0.860	0.870	0.875	**0.900**	0.895	0.895	0.895	0.890
	books	0.750	0.770	0.760	0.780	0.790	0.795	0.785	0.790	**0.800**
	dvd	0.765	0.735	0.770	0.760	0.800	**0.805***	0.785	0.795*	0.795*
	electr.	0.790	0.800	0.800	**0.825**	0.800	0.815	0.805	0.820	0.815
	kitch.	0.760	0.800	0.775	0.800	0.845	**0.860***	0.855	0.840	0.855*

Table 4.2: Accuracy results on the test sets. Bold font marks the best performance for a dataset. * indicates statistical significance of improvement over lasso at $p < 0.05$ using micro sign test for one of our models LSI, GoW and word2vec (underlined).

For the LDA, LSI, sentence, Graph-of-Words (GoW), word2vec regularizers, we perform grid search on the same set of values as ridge and lasso experiments for the ρ, λ_{glas}, λ_{lasso} parameters. In the case we get the same accuracy on the development data, the model with the highest sparsity is selected. For LDA we set the number of topics to 1000 and we keep the 10 most probable words of each topic as a group. For LSI we keep 1000 latent dimensions and we select the 10 most significant words per topic. For the clustering process on word2vec we ran Minibatch-Kmeans for max 2000 clusters. For each word belonging to a cluster, we also keep the top 5 or 10 nearest words so that we introduce overlapping groups. The intuition behind this is that words can be part of multiple "concepts" or topics, thus they can belong to many clusters.

RESULTS In Table 4.2 we report the results of our experiments on the aforementioned datasets, and we distinguish our proposed regularizers LSI, GoW, word2vec with underlining. Our results are inline and confirm that of Yogatama and Smith (2014a) showing the advantages of using structured regularizers in the text categorization task. The group based regularizers perform systematically better than the baseline ones.

We observe that the word2vec clustering based regularizers performs very well - achieving best performance for three out of the ten data sets while it is quite fast

	dataset	no reg.	lasso	ridge	elastic	group lasso				
						LDA	LSI	sentence	GoW	word2vec
20NG	science	100	1	100	63	19	20	86	19	21
	sports	100	1	100	5	60	11	6.4	55	44
	religion	100	1	100	3	94	31	99	10	85
	computer	100	2	100	7	40	35	77	38	18
Sentiment	vote	100	1	100	8	15	16	13	97	13
	movie	100	1	100	59	72	81	55	90	62
	books	100	3	100	14	41	74	72	90	99
	dvd	100	2	100	28	64	8	8	58	64
	electr.	100	4	100	6	10	8	43	8	9
	kitch.	100	5	100	79	73	44	27	75	46

Table 4.3: Fraction (in %) of non-zero feature weights in each model for each dataset: the smaller, the more compact the model.

	Dataset	GoW	word2vec
20NG	science	79	691
	sports	137	630
	religion	35	639
	computer	95	594

Table 4.4: Number of groups.

with regards to execution time as it appears in Table 4.5 (i. e., it is four to ten times faster than the sentence based one).

The LSI based regularization, proposed for the first time in this thesis, performs surprisingly well as it achieves the best performance for three of the ten datasets. This is somehow explained by the fact that this method extracts the inherent dimensions that best represent the different semantics of the documents - as we see as well in the anecdotal examples in Table 4.6, 4.7, 4.8. This method proves as well very fast as it appears in Table 4.5 (i.e. it is three to sixty times faster than the sentence based one).

The GoW based regularization although very fast, did not outperform the other methods (while it has a very good performance in general). It remains to be seen whether a more thorough parameter tuning and community detection algorithm selection would improve further the accuracy of the method.

	Dataset	lasso	ridge	elastic	group lasso				
					LDA	LSI	sentence	GoW	word2vec
20NG	science	10	1.6	1.6	15	11	76	12	19
	sports	12	3	3	7	20	67	5	9
	religion	12	3	7	10	4	248	6	20
	computer	7	1.4	0.8	8	6	43	5	10

Table 4.5: Time (in seconds) for learning with best hyperparameters.

$= 0$	piscataway combination jil@donutso.uucp jamie reading/seeing chambliss left-handedness abilities lubin acad sci obesity page erythromycin bottom
$\neq 0$	and space the launch health for use that medical you space cancer and nasa hiv health shuttle for tobacco that cancer that research center space hiv aids are use theory keyboard data telescope available are from system information space ftp

Table 4.6: Examples with LSI regularizer.

In Table 4.3 we present the feature space sizes retained by each of the regularizers for each dataset. As expected the lasso regularizer sets the vast majority of the features' weights to zero, and thus a very sparse feature space is generated. This fact has as a consequence the significant decrease in accuracy performance. Our proposed structured regularizers managed to perform better in most of the cases, introducing more sparse models compared to the state-of-the-art regularizers.

TIME COMPLEXITY Although certain types of structured regularizers improve significantly the accuracy and address the problem of overfitting, they require a notable amount of time in the learning process.

As seen in Yogatama and Smith (2014b), a considerable disadvantage is the need of search for the optimal hyperparameters: λ_{glas}, λ_{lasso} , and ρ, whereas standard baselines like lasso and ridge only have one hyperparameter and elastic net has two.

Parallel grid search can be critical for finding the optimal set of hyperparameters, since there is no dependency on each other, but again the process can be very expensive. Especially for the case of the sentence regularizer, the process can be

$= 0$	village town edc fashionable trendy trendy fashionable points guard guarding crown title champion champions
$\neq 0$	numbness tingling dizziness fevers laryngitis bronchitis undergo undergoing undergoes undergone healed mankind humanity civilization planet nasa kunin lang tao kay kong

Table 4.7: Examples with word2vec regularizer.

$= 0$	islands inta spain galapagos canary originated anodise advertises jewelry mercedes benzes diamond trendy octave chanute lillienthal
$\neq 0$	vibrational broiled relieving succumb spacewalks dna nf-psychiatry itself commented usenet golded insects alternate self-consistent retrospect

Table 4.8: Examples with Graph-of-Words regularizer.

extremely slow due to two factors. First, the high number of sentences in text data. Second, sentences consist of heavily overlapping groups, that include words reappearing in one or more sentences. On the contrary, as it appears on Table 4.4, the number of clusters in the clustering based regularizers is significantly smaller than that of the sentences - and definitely controlled by the designer - thus resulting in much faster computation. The update of $\mathbf{v}$ still remains time consuming for small datasets, even with parallelization.

Our proposed structured regularizers are considerably faster in reaching convergence, since they offer a smaller number of groups with less overlapping between words. For example, on the computer subset of the 20NG dataset, learning models with the best hyperparameter value(s) for lasso, ridge, and elastic net took 7, 1.4, and 0.8 seconds, respectively, on an Intel Xeon CPU E5-1607 3.00 GHz machine with 4 cores and 128GB RAM. Given the best hyperparameter values the LSI regularizer takes 6 seconds to converge, the word2vec regularizer takes 10 seconds to reach convergence, the Graph-of-Words takes 4 seconds while the sentence regularizer requires 43 seconds. Table 4.5 summarizes required learning time on 20NG datasets.

We also need to consider the time needed to extract the groups. For word2vec, Minibatch K-means requires 15 minutes to cluster the pre-trained vectors by Google. The clustering is executed only once. Getting the clusters of words that belong to the vocabulary of each dataset requires 20 minutes, but can be further optimized. Finding also the communities in the Graph-of-Words approach with the Louvain algorithm, is very fast and requires a few minutes depending on the size and structure of the graph.

In Tables 4.6, 4.7, 4.8 we show examples of our proposed regularizers-removed and -selected groups (in v) in the science subset of the 20NG dataset. Words with weights (in w) of magnitude greater than 10^{-3} are highlighted in red (sci.med) and blue (sci.space).

In this section, we proposed novel structured regularizers for the task of text classification based on word and graph embeddings. We show that our methods significantly outperform standard baselines and in some datasets the state-of-art approaches. Our regularizers, that exploit syntactic and semantic prior knowledge are faster to train and at the same time retain the property of sparsity.

In the next section, we diverge from the group lasso architectures and examine new algorithms for linguistic structured regularization.

4.5 ORTHOGONAL MATCHING PURSUIT FOR TEXT CLASSIFICATION

In text classification, the problem of overfitting arises due to the high dimensionality, making regularization essential. Although classic regularizers provide sparsity, they fail to return highly accurate models. On the contrary, state-of-the-art group-lasso regularizers provide better results at the expense of low sparsity.

In this section, we apply a greedy variable selection algorithm, called Orthogonal Matching Pursuit (OMP), for the text classification task. We also extend standard group OMP by introducing overlapping Group OMP to handle overlapping groups of features. Empirical analysis verifies that both OMP and overlapping GOMP constitute powerful regularizers, able to produce effective and very sparse models. Code and data are available online[11].

11 github.com/y3nko/OMP-for-Text-Classification

4.5.1 *Existing limitations*

The overall high dimensionality of textual data is of major importance in text classification (also known as text categorization), opinion mining, noisy text normalization and other NLP tasks. Since in most cases a high number of words occurs, one can easily fall in the case of overfitting. Regularization remains a key element for addressing overfitting in tasks like text classification, domain adaptation and neural machine translation (Chen and Rosenfeld, 2000; Lu et al., 2016; Barone et al., 2017). Along with better generalization capabilities, a proper scheme of regularization can also introduce sparsity. Recently, a number of text regularization techniques have been proposed in the context of deep learning (Qian et al., 2017; Ma et al., 2017; Zhang et al., 2017).

Apart from L_1, L2 and elastic net, a very popular method for regularizing text classification is group lasso. Yogatama and Smith (2014b) introduced a group lasso variant to utilize groups of words for logistic models. Occasionally though, these groupings are either not available or hard to be extracted. Moreover, no ground truth groups of words exist to validate their quality. Furthermore, group lasso can also fail to create sparse models. Lastly, there has been little work in overlapping group regularization for text, since words can appear in different groups, following the intuition that they can share multiple contexts or topics.

In this work, we apply two greedy variable selection techniques to the text classification task, Orthogonal Matching Pursuit (OMP) and overlapping group Orthogonal Matching Pursuit (GOMP). In the case of GOMP, we build upon work of Lozano et al. (2011), where the authors propose the GOMP algorithm for Logistic Regression for selecting relevant groups of features. More specifically, standard GOMP is based on the assumption that a number of disjoint groups of features are available. Nevertheless, in most cases, these groups are not disjoint. To overcome this problem we extend GOMP to handle overlapping groups of features. We empirically show that both OMP and overlapping GOMP provide highly accurate models, while producing very sparse models compared to group lasso variants.

Our contribution can be summarized in the following novel aspects:

1. apply OMP to text classification;
2. introduce overlapping GOMP, moving from disjoint to overlapping groups;

3. analyze their efficiency in accuracy and sparsity, compared to group lasso variants and state-of-the-art deep learning models.

The rest of this section is organized as follows. Subection 4.5.2 formally introduces the proposed OMP and overlapping GOMP algorithms for the text classification problem. Experimental results are presented in Subsection 4.5.3. We conclude our work in Subsection 4.5.4 by discussing possible future directions.

4.5.2 *Proposed framework*

The vanilla Matching Pursuit (MP) algorithm (Mallat and Zhang, 1993) has its origin in signal processing where it is mainly used in the compressed sensing task. Actually, it approximates the original "signal" iteratively improving the current solution by minimizing the norm of the residual (approximation error). It can also be considered as a forward greedy algorithm for feature selection (dictionary learning problem), that at each iteration uses the correlation between the residual and the candidate features to (greedily) decide which feature to add next. The correlation between the residual and the candidate features is considered to be the length of the orthogonal projection. Then, it subtracts off the correlated part from the residual and performs the same procedure on the updated residual. The algorithm terminates when the residual is lower than a predefined threshold. The final solution is obtained by combining the selected features weighted by their respective correlation values, which are calculated at each iteration.

Orthogonal Matching Pursuit (Pati et al., 1993) is one of the most famous extensions of the matching pursuit algorithm. Similar to MP, OMP can be used for the dictionary learning task where it constitutes a competitive alternative to lasso algorithm. The way it differs from the standard MP is that at every step, all the coefficients extracted so far are updated, by computing the orthogonal projection of the data onto the set of features selected so far. In this way, the newly derived residual is orthogonal to not only the immediately selected feature at the current iteration, but also to all the features that have already been selected. Therefore, OMP never selects the same feature twice. Tropp (2004) provided a theoretical analysis of OMP, which has been generalized by Zhang (2009) on the stochastic noise case.

Algorithm 2 Logistic-OMP

Input: $X = [\mathbf{x}_1, ..., \mathbf{x}_N]^\top \in \mathbb{R}^{N \times d}$, $\mathbf{y} \in \{-1, 1\}^N$, K (budget), ϵ (precision), λ (regularization factor).
Initialize: $\mathcal{I} = \emptyset$, $\mathbf{r}^{(0)} = \mathbf{y}$, $k = 1$
while $|\mathcal{I}| \leqslant K$ **do**
  $j^{(k)} = \arg\max_{j \notin \mathcal{I}} \left| X_j^\top \mathbf{r}^{(k-1)} \right|$
  if $|X_{j^{(k)}}^\top \mathbf{r}^{(k-1)}| \leqslant \epsilon$ **then**
    break
  end if
  $\mathcal{I} = \mathcal{I} \cup \{j^{(k)}\}$
  $\theta^{(k)} = \operatorname{argmin}_\theta \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, \theta, y_i) + \lambda \|\theta\|_2^2 \qquad \text{s.t.} \quad \operatorname{supp}(\theta) \subseteq \mathcal{I}$
  $\mathbf{r}^{(k)} = \frac{1}{1+\exp\{-X\theta^{(k)}\}} - \mathbb{1}_{\{\mathbf{y}\}}$
  $k \mathrel{+}= 1$
end while
return $\theta^{(k)}, \mathcal{I}$

In the following part, we explain the main steps of the logistic OMP algorithm in detail. Given a training set, we define $X = [\mathbf{x}_1, ..., \mathbf{x}_N]^\top \in \mathbb{R}^{N \times d}$ to be the (dictionary) matrix of features (or variables) vectors, with each column X_j to represent a feature, $\mathbf{f}_j \in \mathbb{R}^N$. Let also $\mathbf{y} = [y_1, \ldots, y_N]^\top$ denote the response vector. For any set of indices $\mathcal{I}$, let $X_\mathcal{I}$ denote a subset of features from X, such that feature $\mathbf{f}_j$ is included in $X_\mathcal{I}$ if $j \in \mathcal{I}$. Thus, $X_\mathcal{I} = \{\mathbf{f}_j, j \in \mathcal{I}\}$, with the columns $\mathbf{f}_j$ to be arranged in ascending order.

OMP starts by setting the residual equal to the response vector, $\mathbf{r}^{(0)} = \mathbf{y}$, assuming that the set of indices $\mathcal{I}$ (contains the indices of the active features) is initially empty. At each iteration k, OMP activates the feature that has the maximum correlation with the residual $\mathbf{r}^{(k-1)}$ (calculated in the previous step):

$$j^{(k)} = \arg\max_{j \notin \mathcal{I}} \left| X_j^\top \mathbf{r}^{(k-1)} \right|. \tag{4.21}$$

Then, we incorporate the index $j^{(k)}$ to the set $\mathcal{I}$, i. e., $\mathcal{I} = \mathcal{I} \cup \{j^{(k)}\}$. Afterwards, we apply the ordinary logistic regression (LR) by considering only the *active* features. More specifically, we get the optimal coefficients by minimizing the negative log likelihood along with an L2 penalty term:

$$\theta^{(k)} = \operatorname*{argmin}_\theta \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, \theta, y_i) + \lambda \|\theta\|_2^2,$$

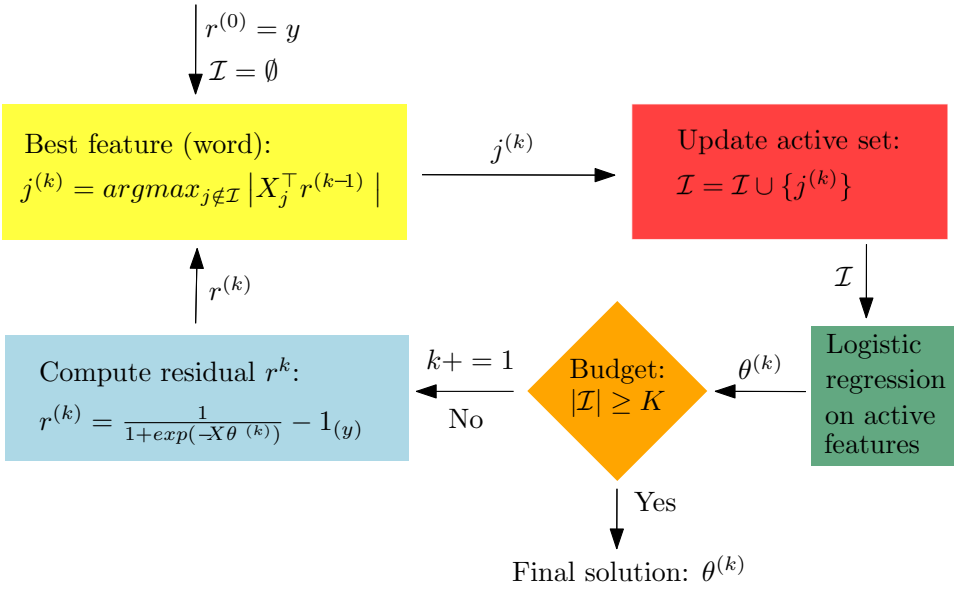


Figure 4.2: OMP pipeline where $X \in \mathbb{R}^{N \times d}$ is the design matrix, $y \in \mathbb{R}^N$ is the response vector, K is our budget and $\mathcal{I}$ the set of active features.

$$\text{s.t.} \quad \text{supp}(\theta) \subseteq \mathcal{I} \tag{4.22}$$

where $\text{supp}(\theta) = \{j : \theta_j \neq 0\}$. Roughly speaking, the values of the coefficients correspond to inactive features (indices) forced to be equal to zero.

Finally, we calculate the updated residual:

$$\mathbf{r}^{(k)} = \frac{1}{1 + \exp\{-X\theta^{(k)}\}} - \mathbb{1}_{\{y\}}, \tag{4.23}$$

where $\mathbb{1}_{\{y\}} \triangleq \mathbb{1}\{y_i \in \{1\}, \forall i \in \{1, \ldots, n\}\}$ indicates if instance $\mathbf{x}_i$ belongs to class 1 or not. We repeat the process until the residual becomes smaller than a predefined threshold, $\epsilon \geqslant 0$, or a desired number of active features, K (budget), has been selected. Through our empirical analysis we set $\epsilon = 0$, examining only the number of active features. An overview of logistic-OMP is given in Alg. 2. A detailed analysis of the algorithm's complexity is provided by Tropp and Gilbert (2007).

4.5.2.1 *Overlapping Group OMP*

The Group OMP (GOMP) algorithm was originally introduced by Swirszcz et al. (2009) for linear regression models, and extended by Lozano et al. (2011) in order

Algorithm 3 Logistic Overlapping GOMP

Input: $X = [x_1, ..., x_N]^\top \in \mathbb{R}^{N \times d}$, $y \in \{-1, 1\}^N$, $\{G_1, \ldots, G_J\}$ (group structure), K (budget), ϵ (precision), λ (regularization factor).
Initialize: $\mathcal{I} = \emptyset$, $r^{(0)} = y$, $k = 1$

while $|\mathcal{I}| \leqslant K$ **do**
  $j^{(k)} = \arg\max_j \frac{1}{|G_j|} \left\| X_{G_j}^\top r^{(k-1)} \right\|_2^2$
  if $\left\| X_{G_{j^{(k)}}}^\top r^{(k-1)} \right\|_2^2 \leqslant \epsilon$ **then**
    break
  end if
  $\mathcal{I} = \mathcal{I} \cup \{G_{j^{(k)}}\}$
  for $i = 1$ **to** J **do**
    $G_i = G_i \setminus G_{j^{(k)}}$
  end for
  $\theta^{(k)} = \operatorname{argmin}_\theta \sum_{i=1}^N \mathcal{L}(x_i, \theta, y_i) + \lambda \|\theta\|_2^2 \qquad \text{s.t.} \quad \operatorname{supp}(\theta) \subseteq \mathcal{I}$
  $r^{(k)} = \frac{1}{1+\exp\{-X\theta^{(k)}\}} - \mathbb{1}_{\{y\}}$
  $k \mathrel{+}= 1$
end while
return $\theta^{(k)}, \mathcal{I}$

to select groups of variables in logistic regression models. Following the notion of group lasso, GOMP utilizes prior knowledge about groups of features in order to penalize large weights in a collective way. Given that we have words sharing some properties, we can leverage these grouping for regularization purposes. Figure 4.2 illustrates the complete text classification pipeline, including the GOMP component, needed for the regularization part.

Similar to Lozano et al. (2011), let us assume that a natural grouping structure exists within the variables consisting of J groups $X_{G_1}, \ldots, X_{G_J}$, where $G_i \subset \{1, \ldots, d\}$, and $X_{G_i} \in \mathbb{R}^{N \times |G_i|}$. The standard GOMP algorithm also assumes that the groups are disjoint, $G_i \cap G_j = \emptyset$ for $i \neq j$. We will remove this assumption later on, by proposing the overlapping GOMP algorithm that is able to handle overlapping groups of features. GOMP operates in the same way with OMP but instead of selecting a single feature, it selects a group of features with the maximum correlation between them and the residual:

$$j^{(k)} = \arg\max_j \left\| X_{G_j}^\top r^{(k-1)} \right\|_2^2. \tag{4.24}$$

In the case where the groups are not orthonormalized (i. e., $X_{G_j}^\top X_{G_j} = \mathcal{I}_{G_j}$, where $\mathcal{I}_{G_j}$ is the identity matrix of size $\mathbb{R}^{|G_j| \times |G_j|}$), we select the best group based on the next criterion:

$$j^{(k)} = \arg\max_j \left| \left(\mathbf{r}^{(k-1)} \right)^\top X_{G_j} (X_{G_j}^\top X_{G_j})^{-1} X_{G_j}^\top \mathbf{r}^{(k-1)} \right|. \tag{4.25}$$

During our empirical analysis, we have noticed that the aforementioned criteria benefit large groups. This becomes apparent especially in the case where the size of the groups is not balanced. In this way, groups with a large number of "irrelevant" features are highly probable to be added. For instance, it is more probable to add a group that consists of 2 good features and 100 bad features, instead of a group that contains only 2 good features. To deal with situations like this one, we consider the average correlation between the group's features and the residual:

$$j^{(k)} = \arg\max_j \frac{1}{|G_j|} \left\| X_{G_j}^\top \mathbf{r}^{(k-1)} \right\|_2^2. \tag{4.26}$$

Overlapping GOMP extends the standard GOMP in the case where the groups of indices are overlapping, i. e., $G_i \cap G_j \neq \emptyset$ for $i \neq j$. The main difference with GOMP is that each time a group becomes active, we remove its indices from each inactive group: $G_i = G_i \setminus G_{j^{(k)}}, \quad \forall i \in \{1, \ldots, J\}$. In this way, the theoretical properties of GOMP hold also in the case of the overlapping GOMP algorithm. A sketch of the overlapping GOMP is shown in Alg. 3.

Figure 4.2 illustrates schematically the pipeline of OMP.

4.5.3 *Experiments*

Next, we present the data, setup and results of our empirical analysis on the text classification task. We also describe the algorithms used as baselines for comparison. To demonstrate the capabilities of the proposed OMP and GOMP algorithms on the text classification task, we considered ten well-known datasets. In total, we tested our method on ten binary classification tasks.

DATASETS We use the same datasets as the previous section for comparison purposes. Table 4.1 summarizes statistics about the aforementioned datasets used in our experiments. We choose small datasets intentionally, like Yogatama and Smith (2014b), so that we can observe the regularization effect clearly.

	Dataset	no reg	lasso	ridge	elastic	OMP	Group lasso regularizers					GOMP
							LDA	LSI	sen	GoW	w2v	
20NG	science	0.946	0.916	0.954	0.954	0.964*	**0.968**	**0.968***	0.942	0.967*	**0.968***	0.953*
	sports	0.908	0.907	0.925	0.920	0.949*	0.959	0.964*	**0.966**	0.959*	0.946*	0.951*
	religion	0.894	0.876	0.895	0.890	0.902*	0.918	0.907*	**0.934**	0.911*	0.916*	0.902*
	computer	0.846	0.843	0.869	0.856	0.876*	0.891	0.885*	0.904	0.885*	**0.911***	0.902*
Sentiment	vote	0.606	0.643	0.616	0.622	0.684*	0.658	0.653	0.656	0.640	0.651	**0.687***
	movie	0.865	0.860	0.870	0.875	0.860*	**0.900**	0.895	0.895	0.895	0.890	0.850
	books	0.750	0.770	0.760	0.780	0.800	0.790	0.795	0.785	0.790	0.800	**0.805***
	dvd	0.765	0.735	0.770	0.760	0.785	0.800	0.805*	0.785	0.795*	0.795*	**0.820***
	electr.	0.790	0.800	0.800	0.825	**0.830**	0.800	0.815	0.805	0.820	0.815	0.800
	kitch.	0.760	0.800	0.775	0.800	0.825	0.845	**0.860***	0.855	0.840	0.855*	0.830

Table 4.9: Accuracy on the test sets. Bold font marks the best performance for a dataset, while * indicates statistical significance at $p < 0.05$ using micro sign test against lasso. For GOMP, we use w2v clusters and add all unigram features as individual groups.

EXPERIMENTAL SETUP In our setup, as features we use unigram frequency concatenated with an additional bias term. We reproduce standard regularizers like lasso, ridge, elastic and state-of-the-art structured regularizers like sentence, LDA, GoW and w2v groups (Skianis et al., 2016a) as baselines and compare them with the proposed OMP and GOMP.

PARAMETER TUNING All the hyperparameters are tuned on the development dataset, using accuracy for evaluation. For lasso and ridge regularization, we choose λ from $10\{^{-2}, 10^{-1}, 1, 10, 10^2\}$. For elastic net, we perform grid search on the same set of values as ridge and lasso experiments for λ_{ridge} and λ_{lasso}. For group lasso, OMP and GOMP regularizers, we perform grid search on the same set of parameters as ridge and lasso experiments. In the case we get the same accuracy on the development data, the model with the highest sparsity is selected. In GOMP we considered all individual features as separate groups of size one, along with the w2v groups. Last but not least, in both OMP and GOMP the maximum number of features, K(budget), is set to 2000.

RESULTS Table 4.9 reports the results of our experiments on the aforementioned datasets. The empirical results reveal the advantages of using OMP or GOMP for regularization in the text categorization task. The OMP regularizer performs systematically better than the baseline ones. More specifically, OMP outperforms the lasso, ridge and elastic net regularizers in all datasets, as regards to the accuracy.

	Dataset	no reg	lasso	ridge	elastic	OMP	Group lasso regularizers					GOMP
							LDA	LSI	sen	GoW	w2v	
20NG	science	100	**1**	100	63	2.7	19	20	86	19	21	5.8
	sports	100	**1**	100	5	1.8	60	11	6.4	55	44	7.7
	religion	100	**1.1**	100	3	1.5	94	31	99	10	85	1.5
	computer	100	1.6	100	7	**0.6**	40	35	77	38	18	4.9
Sentiment	vote	100	**0.1**	100	8	5	15	16	13	97	13	1.5
	movie	100	1.3	100	59	**0.9**	72	81	55	90	62	2.3
	books	100	**3.3**	100	14	4.6	41	74	72	90	99	8.3
	dvd	100	**2**	100	28	2.8	64	8	8	58	64	9
	electr.	100	**4**	100	6	6.3	10	8	43	8	9	12
	kitch.	100	4.5	100	79	**4.3**	73	44	27	75	46	6.5

Table 4.10: Model sizes (percentages of non-zero features in the resulting models). Bold for best, blue for best group.

science	lasso	orbit, space, contribute, funding, landing
	OMP	space, orbit, moon, planets, scientifically

Table 4.11: Largest positive weights in lasso and OMP for the science subset of 20NG.

At the same time, the performance of OMP is quite close or even better to that of structured regularizers. Actually, in the case of electronics data, the model produced by OMP is the one with the highest accuracy. On the other hand, the proposed overlapping GOMP regularizer outperforms all the other regularizers in 3 out of 10 datasets.

Another important observation is how GOMP performs with different types of groups. GOMP only requires some "good" groups along with single features in order to achieve good accuracy. Smaller groups provided by LDA, LSI and w2v clusters provide a good solution and also fast computation, while others (GoW communities) can produce similar results with slower learning times. This phenomenon can be attributed to the different structure of groups. While LDA and LSI have a large number of groups with small number of features in them (1000 groups, 10 words per group), w2v clusters and GoW communities consist of smaller number of groups with larger number of words belonging to each group. Nevertheless, we have reached the conclusion that the selection of groups is not crucial for the general performance of the proposed GOMP algorithm.

Table 4.10 shows the sparsity sizes of all the regularizers we tested. As it becomes apparent, both OMP and GOMP yield super-sparse models, with good

	Dataset	CNN (20eps)	FastText (100eps)	Best OMP or GOMP	Best Lasso
20NG	science	0.935	0.958	0.964	**0.968**
	sports	0.924	0.935	0.951	**0.966**
	religion	**0.934**	0.898	0.902	**0.934**
	computer	0.885	0.867	0.902	**0.911**
Sentiment	vote	0.651	0.643	**0.687**	0.658
	movie	0.780	0.875	0.860	**0.900**
	books	0.742	0.787	**0.805**	0.800
	dvd	0.732	0.757	**0.820**	0.805
	electr.	0.760	0.800	**0.830**	0.820
	kitch.	0.805	0.845	0.830	**0.860**

Table 4.12: Comparison in test accuracy with state-of-the-art classifiers: CNN (Kim, 2014), FastText (Joulin et al., 2017) with no pre-trained vectors. The proposed OMP and GOMP algorithms produce the highest accurate model in 4 out of 10 datasets.

generalization capabilities. More specifically, OMP produces sparse spaces similar to lasso, while GOMP keeps a significantly lower number of features compared to the other structured regularizers. In group regularization, GOMP achieves both best accuracy and sparsity in two datasets (vote & books), while group lasso only in one (sports).

In Table 4.11 we demonstrate the ability of OMP to produce more discriminative features compared to lasso by showing the largest weights and their respective term.

Finally, in Table 4.12 we compare state-of-the-art group lasso classifiers with deep learning architectures (Kim, 2014) with Dropout (Srivastava et al., 2014) for regularization and FastText (Joulin et al., 2017). We show that group lasso regularizers with simple logistic models remain very effective. Nevertheless, adding pre-trained vectors in the deep learning techniques and performing parameter tuning would definitely increase their performance against our models, but with a significant cost in time complexity.

SPARSITY VS ACCURACY Figure 4.3 visualizes the accuracy vs. sparsity for all datasets and all classifiers. We do that in order to identify the best models, by both metrics. The desirable is for classifiers to belong in the top right corner,

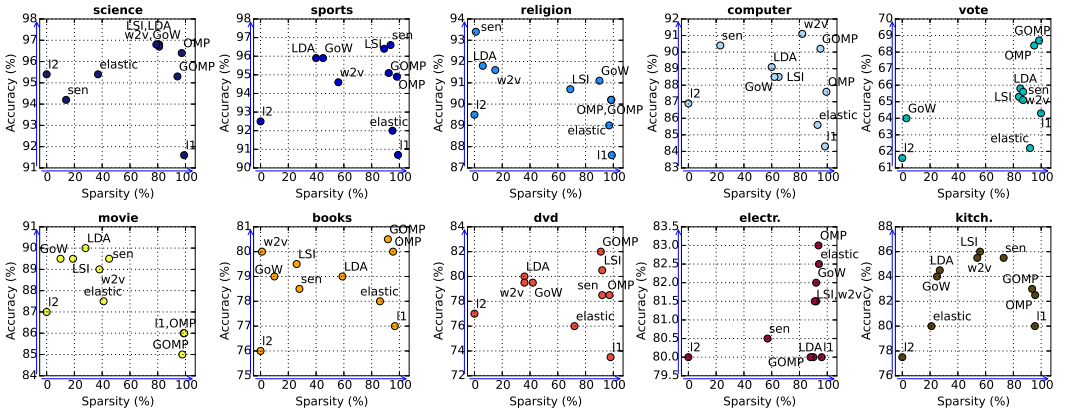


Figure 4.3: Accuracy vs sparsity on the test sets. Regularizers close to the top right corner are preferred.

offering high accuracy and high sparsity at the same time. We observe that OMP and GOMP tend to belong in the right parts of the plot, having very high sparsity, often comparable to the aggressive lasso, even when they do not achieve the best accuracies.

NUMBER OF ACTIVE FEATURES (ATOMS) In both OMP and GOMP algorithms, the maximum desired number of active features (K, budget) was used as stopping criterion. For instance, by setting K = 1000, the proposed methods return the learned values that correspond to the first $\{100, 200, \ldots, 1000\}$ features, respectively. Thus, we exploit the feedforward feature selection structures of OMP and GOMP.

Figure 4.4 presents the number of active features versus accuracy in the development subsets of the 20NG dataset. It can be easily observed that after selecting 1000 active atoms, the accuracy stabilizes or even drops (overfitting problem). For instance, the best number of active features are: i) science: 700, ii) sports: 1100, iii) religion: 400 and iv) computer: 1500. The reason for selecting K = 2000 as the number of features to examine was to provide a sufficient number for OMP to reach a good accuracy while providing a super-sparse solution comparable to lasso.

TIME COMPLEXITY Although certain types of group lasso regularizers perform well, they require a notable amount of time in the learning process.

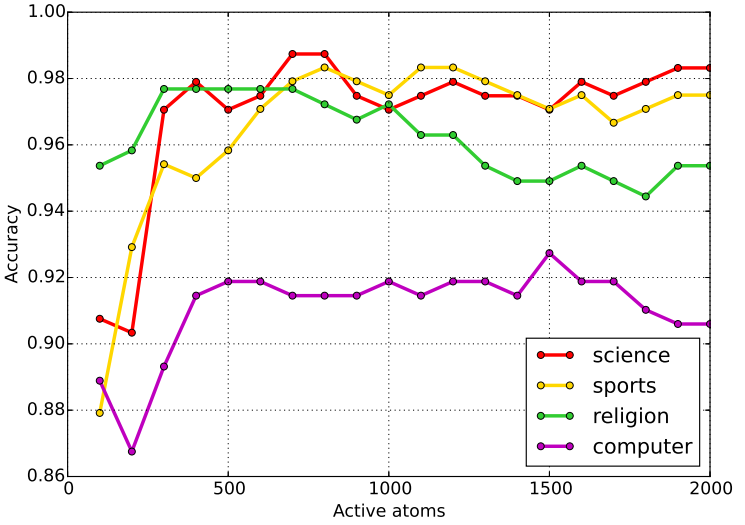


Figure 4.4: Accuracy vs. number of active atoms/features for OMP on 20NG data.

OMP offers fast learning time, given the hyperparameter values and the number of atoms. For example, on the computer subset of the 20NG dataset, learning models with the best hyperparameter value(s) for lasso, ridge, and elastic net took 7, 1.4, and 0.8 seconds, respectively, on a 4-core 3.00GHz CPU. On the other hand, OMP requires only 4 seconds for training, making it even faster than lasso, while providing a sparser model.

GOMP can have very slow learning time when adding the features as groups individually. This is due to the large number of groups that GOMP needs to explore in order to extract the most "contributing" ones. If we consider GOMP without the individual features as groups, then the learning process becomes faster, with a clear decreasing effect on accuracy. In general, groups need to be well structured for GOMP to manage to surpass OMP and other state-of-the-art group lasso regularizers.

The advantages of the proposed methods are: (1) OMP requires no prior structural knowledge, (2) producing more discriminative features and (3) fast with relatively small number of dimensions.

Moreover, our implementation compared to the one of Lozano et al. (2011), provides the advantage of storing the weights and not having to recompute the whole matrices from scratch.

In the drawbacks of the methods: (1) OMP and GOMP are greedy algorithms, thus GOMP gets slow when we add the features as individual groups and (2) groups need to be “good".

4.5.4 *Summary*

In this part of the thesis, we introduced OMP and GOMP algorithms on the text classification task. An extension of the standard GOMP algorithm was also proposed, which is able to handle overlapping groups. The main advantages of both OMP and GOMP compared to other regularization schemes are their simplicity (greedy feedforward feature selection) and ability to produce accurate models with good generalization capabilities. We have shown that the proposed classifiers outperform standard baselines, as well as state-of-art structured regularizers in various datasets. Similar to Mosci et al. (2010), Yen et al. (2017), and Xie et al. (2017), our empirical analysis validates that regularization remains a highly important topic, especially for deep learning models (Roth et al., 2017).

4.6 CONCLUSION & FUTURE WORK

CONCLUSION In this chapter we proposed new types of structured regularizers to improve not only the accuracy but also the efficiency of the text categorization task. We mainly focused on how to find and extract semantic and syntactic structures that lead to sparser feature spaces and therefore to faster learning times. Overall, our results demonstrate that linguistic prior knowledge in the data can be used to improve categorization performance for baseline Bag-of-Words models, by mining inherent structures.

Furthermore, we introduced OMP and GOMP algorithms on the text classification task. An extension of the standard GOMP algorithm was also proposed, which is able to handle overlapping groups. The main advantages of both OMP and GOMP compared to other regularization schemes are their simplicity (greedy feedforward feature selection) and ability to produce accurate models with good generalization capabilities, while introducing larger sparsity compared to group-lasso variants.

FUTURE WORK In the future, we would like to involve a more thorough investigation on how to create and cluster graphs, i. e., covering weighted and/or signed cases. Finding better clusters in the word2vec space is also a critical part. This is not only restricted in finding the best number of clusters but what type of clusters we are trying to extract. Gaussian Mixture Models (McLachlan and Basford, 1988) could be applied in order to capture overlapping groups at the cost of high complexity. The intuition behind this idea is that words can be part of multiple and diverse topics. Furthermore, topical word embeddings (Liu et al., 2015) can be considered for regularization. This approach could enhance the regularization on topic specific datasets.

As mentioned previously, groups are not always specified in advance or hard to extract. Especially in environments involving text. To address this problem, we plan to extend our work by learning automatically the groups with Simultaneous Orthogonal Matching Pursuit (Szlam et al., 2012). Another interesting future direction would be to additionally penalize features inside the groups, similarly to sparse group lasso. Moreover, it would be highly interesting to examine the theoretical properties of overlapping GOMP. Similar to Mosci et al. (2010), Yen et al. (2017), and Xie et al. (2017), our empirical analysis validates that regularization remains a highly important topic and should be further studied for current state-of-the-art deep learning approaches (Roth et al., 2017).

Modern neural network architectures are commonly augmented with an attention mechanism, which tells the network where to look within the input in order to make the next prediction. Attention augmented architectures have been successfully applied to many and diverse areas ranging from machine translation (Bahdanau et al., 2015; Luong et al., 2015), to speech recognition (Chorowski et al., 2015), image caption generation (Xu et al., 2015), textual entailment (Rocktäschel et al., 2016; Martins and Astudillo, 2016), and sentence summarization (Rush et al., 2015). In our current work (Skianis et al., 2019b), we study group lasso as an attention mechanism, which will be easily fitted to deep learning models for structured regularization.

5

SETS & DISTANCES IN WORD EMBEDDINGS

Word embeddings have opened a new path in creating novel approaches for addressing traditional problems in the NLP domain. Their use has lead to state-of-the-art results in numerous tasks like document similarity, text classification or in general language modelling. However, using word embeddings to compare text documents remains a relatively unexplored topic – with Word Mover's Distance (WMD) being the prominent tool used so far. In this chapter, we present a variety of tools that can further improve the computation of distances between documents. We demonstrate that simple tricks, like stop-word removal, cross document-topic comparison based on our proposed linguistic groups and alternative convex metric learning techniques constitute powerful tools that can boost WMD towards measuring distance between documents.

Apart from Word Mover's Distance, one can utilize bipartite matching and network max-flow, to compute distances between distributions. Assuming we have access to textual data objects and label information, we can train a model based on the previous approaches. In several domains, as well as NLP, data objects can be decomposed into sets of simpler objects. It is then natural to represent each object as the set of its components or parts. Many conventional machine learning algorithms are unable to process this kind of representations, since sets may vary in cardinality and elements lack a meaningful ordering. In this part, we present a new neural network architecture called RepSet) that can handle examples that are represented as sets of vectors. The proposed model computes the correspondences between an input set and some hidden sets by solving a series of network flow problems. This representation is then fed to a standard neural network architecture to produce the output. The architecture allows end-to-end gradient-based learning. We demonstrate the proposed model on the text classification task and we show that the proposed neural network achieves performance better or comparable to state-of-the-art algorithms.

5.1 INTRODUCTION

Measuring distance between documents has always been a key component in many natural language processing tasks, such as document classification (Bigi, 2003), machine translation (Zhang et al., 2016), question answering (Brokos et al., 2016) and text generation (Chen et al., 2018). In order to quantify the mutual information that two objects can carry, we need to derive a similarity real-valued function. For text, the objects can be two text documents, forming reviews, news e.t.c. Nevertheless, the task can present various difficulties, making it not trivial; whether two documents are similar or not, is not always clear and may vary from application to application.

Following a naive, but effective in many cases, assumption, previous similarity measures that make use of the vector space model (Salton et al., 1975), were treating words in a document as if they were independent to each other. On the contrary, the distributional hypothesis (Harris, 1954), stated that words that co-occur in similar contexts and frequently, tend to have similar meanings and share common semantics.

With the rise of neural networks and deep learning methods in the natural language processing community (Bengio et al., 2003; Collobert and Weston, 2008), word embeddings (Mikolov et al., 2013b) have had a huge impact in numerous tasks. Apart from constituting the most popular input for CNN (Kim, 2014) and LSTM (Johnson and Zhang, 2016), word embeddings have been used to compute similarity between documents that might not carry any identical words.

5.2 RELATED WORK

Following the idea of using Earth Mover's Distance (EMD) to measure document distance (Tao et al., 2012), recent work presented Word Mover's Distance (Kusner et al., 2015), a method for measuring distances between documents in the word vector space. Word Mover's Distance is based on solving the Monge-Kantorovich problem, which is essentially finding the optimal transport (Villani, 2008) between two distributions. Figure 5.1 shows a schematic illustration of the WMD metric.

Moving forward, a supervised version of Word Mover's Distance has been introduced (Huang et al., 2016), which employs metric learning techniques when label knowledge exists. Their work is based on regularized optimal transport by

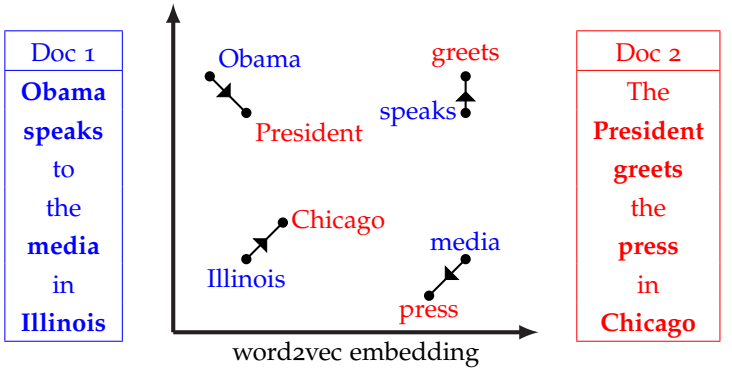


Figure 5.1: The Word Mover's Distance by Kusner et al. (2015).

Cuturi (2013), who presented an $\mathcal{O}(n^2)$ algorithm. Their approaches have shown unprecedented results in the task of kNN text classification.

Although Word Mover's Distance is a powerful method for comparing two text documents, it can fall into the case where a word is very common and thus not contributing to measuring the distance. Moreover the exact computation of WMD scales at $\mathcal{O}(n^3)$, making it prohibitive for large collections of documents with big vocabularies. Kusner et al. (2015) addressed this problem with a much faster variant, the Relaxed WMD (RWMD), which is a lower bound of the exact solution of WMD. Later, Atasu et al. (2017) provided a linear complexity algorithm for computing Relaxed Word Mover's Distance.

LEARNING SETS As we care for learning hidden representations, we examine current state-of-the-art approaches, suitable not only for the task of text classification, which are close to our proposed approach. In their seminal work, Rubner et al. (2000) presented Earth Mover's Distance (EMD) as a Metric for Image Retrieval. The EMD is based on the minimal cost that must be paid to transform one distribution into the other, in a precise sense, and was first proposed for certain vision problems by Peleg et al. (1989). Kusner et al. (2015) built on this idea, and used EMD for distance computation with word embeddings. Their method, called Word Mover's Distance, was the first way to compute distance between two text documents, given a word vector space. Later, a Supervised version of Word Mover's Distance(Huang et al., 2016) was presented, that enabled to utilize label information via metric learning (Xing et al., 2003; Goldberger et al., 2005; Yang and Jin, 2006).

Besides EMD for text similarity, several kernels between sets of vectors have been proposed in the past to enable kernel methods (e. g., SVMs) to handle unordered sets.

Most of these kernels estimate a probability distribution on each set of vectors, and then derive their similarity using distribution-based comparison measures such as Fisher kernels (Jaakkola and Haussler, 1999), probability product kernels (Jebara et al., 2004; Lyu, 2005) and the classical Bhattacharyya similarity measure (Kondor and Jebara, 2003). Furthermore, there are also kernels that map the vectors of each set to multi-resolution histograms, and then in order to find an approximate correspondence between the two sets of vectors, they compare the histograms with a weighted histogram intersection measure (Grauman and Darrell, 2007b; Grauman and Darrell, 2007a). Such kernels have been applied to different tasks such as to the problem of graph classification (Nikolentzos et al., 2017a). Although very effective in several tasks, these kernel-based approaches suffer from high computational complexity. In most cases, the complexity of computing kernels between sets is quadratic in the number of vectors per set, while when dealing with classification problems, the complexity of optimizing the SVM classifier is quadratic in the number of training samples.

The past years witnessed a surge of interest in the area of neural networks for sets (Qi et al., 2017a; Zaheer et al., 2017; Li et al., 2018; Vinyals et al., 2015; Rezatofighi et al., 2017; Rezatofighi et al., 2018). These networks served mainly as the answer to computer vision problems such as the automated classification of point clouds. Although conceptually simple, the proposed architectures have achieved state-of-the-art results on many tasks. PointNet (Qi et al., 2017a) and DeepSets (Zaheer et al., 2017) transform the vectors of the sets (i. e., using several layers) into new representations. They then apply some permutation-invariant function to the emerging vectors to generate representations for the sets. PointNet uses max-pooling to aggregate information across vectors, while DeepSets adds up the representations of the vectors. The representation of the set is then passed on to a standard architecture (e. g., fully connected layers, nonlinearities, etc). PointNet++ Qi et al., 2017b and SO-Net (Li et al., 2018) apply PointNet hierarchically in order to better capture local structures. Two other recent works employ neural networks to learn the parameters of the likelihood of each set (Rezatofighi et al., 2017; Rezatofighi et al., 2018). Vinyals *et al.* treat in Vinyals et al. (2015) unordered sets as ordered sequences and apply RNN models to them. However, they show

that the output of the network is highly dependent on the order of the elements of the set.

5.3 BACKGROUND

In this section, we set the background and preliminaries on which we based our work on boosting Word Mover's Distance and learning set representations via bipartite matching.

5.3.1 *Word Mover's Distance*

Let's assume that we have access to a word embedding matrix $X \in \mathbb{R}^{n \times d}$, for example pre-trained vectors by Mikolov et al. (2013b), for a finite size vocabulary of n words. $x_i \in \mathbb{R}^d$ represents the embedding of the i-th word in a d-dimensional space. We assume that text documents are represented as normalized Bag-of-Words vectors, $d \in \mathbb{R}^n$. Word Mover's Distance tries to embody the semantic similarity between individual word pairs into the document distance metric. One such measure of word dissimilarity is naturally provided by their Euclidean distance in the word embedding space.

The Word Mover's Distance (WMD), utilizes this property of word2vec embeddings. We represent text documents as a weighted point cloud of embedded words. The distance between two text documents A and B is the minimum cumulative distance that words from document A need to travel to match exactly the point cloud of document B.

As Word Mover's Distance consists of multiple components, several improvement suggestions can be done. We observe that many tools can be of service, such as:

(a) dimensionality reduction, where some of the dimensions are actually useful making the computation faster,

(b) POS-tagging matching, matching by POS-tags,

(c) stopword removal, which stopwords to remove

(d) topic modelling, adding words that belong in the same topic

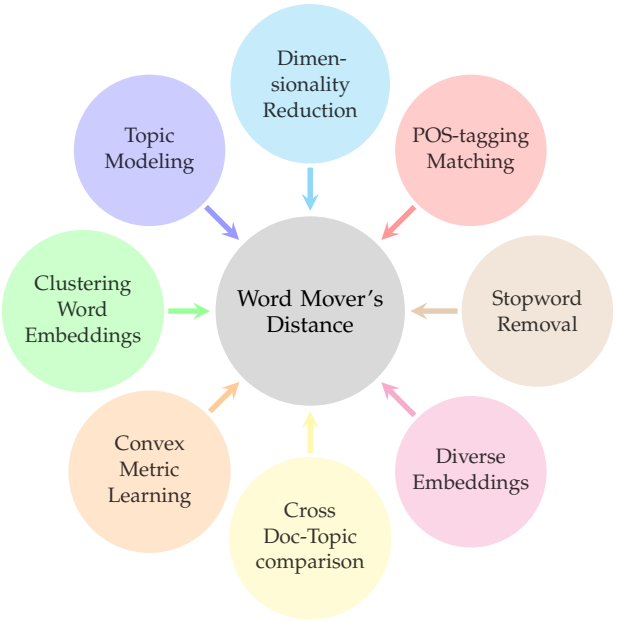


Figure 5.2: Tools to boost WMD.

(e) clustering word embeddings,

(f) cross doc-topic comparison, adding neighbor words

(g) different embeddings, like the recently introduced Poincare (Nickel and Kiela, 2017), ELMo (Peters et al., 2018) and state-of-the-art BERT embeddings (Devlin et al., 2018),

(h) metric learning, assuming we have access to label information.

An illustration of the possible enhancing tools is shown in Figure 5.2.

TOPIC EXTRACTION In order to extract topics from the word embedding space, clustering in word vectors (Skianis et al., 2016a) may be applied to obtain groups of words, which then can be used for regularizing logistic models in text classification. Kim et al. (2017) utilizes Word Mover's Distance to identify related words when no direct matches are found between a query and a document. In recent work, a topical distance approach (Witt et al., 2016) was attempted using word embeddings, by iteratively picking words from a vocabulary that closes the topical gap between documents.

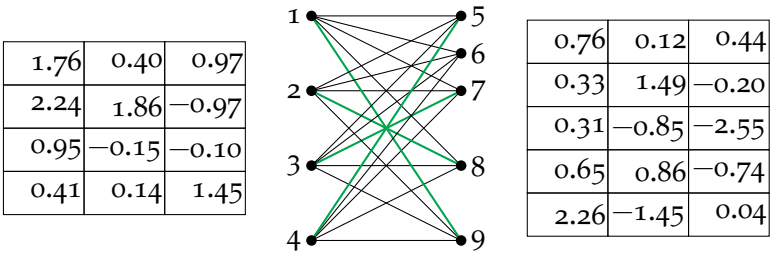


Figure 5.3: Example of a bipartite graph between 2 sets with dimension 3. Green color indicates that an edge belongs to the matching. The total cost is 8.93.

METRIC LEARNING Metric or distance learning is a field covering both supervised and unsupervised techniques (Yang and Jin, 2006). As an extension of Word Mover's Distance, Supervised Word Mover's Distance (Huang et al., 2016) was presented, a method which utilized Neighborhood Component Analysis (NCA) (Goldberger et al., 2005) along with word embeddings and documents labels. Apart from NCA, there exists a plethora of popular methods for generalized Euclidean metric learning. Information-Theoretic Metric Learning (ITML) (Davis et al., 2007) learns a metric by minimizing a KL-divergence subject to generalized Euclidean distance constraints.

5.3.2 *Weighted Bipartite Matching*

Apart from Earth Mover's Distance, which is based on finding the optimal transport, different techniques can be used for computing distance between two distributions. For example, if we model the two sets as a bipartite graph, we can solve the pairing problem with bipartite matching.

The maximum weighted bipartite matching is one of the most well–studied problems in combinatorial optimization. The input of the problem is a weighted bipartite graph $G = (V, E)$. The set of nodes V can be decomposed into two disjoint sets V_1 and V_2, i. e., $V = V_1 \cup V_2$. Every edge $e \in E$ connects a vertex in V_1 to one in V_2. A matching M is a subset of edges such that each node in V appears in at most one edge in M. The optimal solution to the problem can be interpreted as the similarity between the two node sets V_1 and V_2. A bipartite graph has a natural representation as a rectangular $|V_1| \times |V_2|$ matrix where the ij^{th} component is equal to the weight of the edge between the i^{th} element of V_1 and the j^{th} element of V_2 if that edge exists, otherwise equal to 0. Figure 5.3 illustrates a weighted bipartite

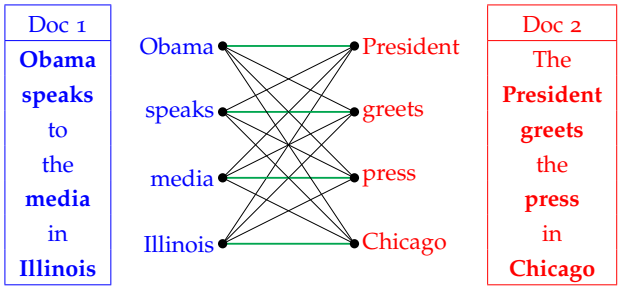


Figure 5.4: Bipartite Graph Matching for two documents.

graph along with the optimal matching M. Green edges belong to the matching M.

5.3.3 *Weighted Bipartite Matching for Text*

We present an example for weighted bipartite matching between two text documents. The nodes in the bipartite graph represent the words of the two documents, and on top of the edges we can have as weights the similarity of their corresponding vectors. We see that there is one-to-one matching only, making it much faster than the Word Mover's Distance approach. A schematic illustration is shown in Figure 5.4.

5.4 BOOSTING WORD MOVER'S DISTANCE

Our contribution is focused on studying the contribution of the following three tools in the computation of WMD: vocabulary trimming with stopword removal, cross document-topic comparison and convex metric learning methods.

First, by selecting specific stopwords (Manning et al., 1999) we see that they play a significant role in the distance computation process. Next, with utilizing cross document-topic comparison, we aim to make the comparison of two documents more meaningful by utilizing additional neighbour words. Finally, in order to boost the supervised version of WMD (S-WMD), we apply two state-of-the-art convex metric learning algorithms, Large Margin Nearest Neighbors (LMNN)

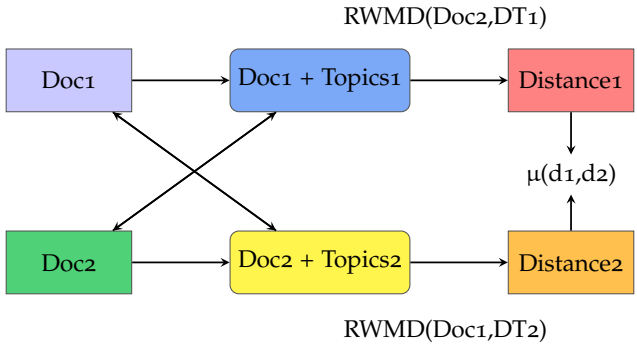


Figure 5.5: Cross document-topic comparison schematic. DT1 stands for adding Doc1 words with Topics1.

(Weinberger and Saul, 2009), as well as Maximally Collapsing Metric Learning (MCML) (Globerson and Roweis, 2006).

5.4.1 *Stopword Removal*

Vocabulary trimming or pruning can help us to get rid of insignificant words, making the Relaxed WMD faster, while producing a better comparison between documents. In this way, we can make the "travel cost" needed from one document to another cheaper, faster and more effective, as the remaining words (after trimming) are the actual words that contribute to the meaning. Stopwords are in general category independent and thus the first that one could consider irrelevant. In recent work by Schofield et al. (2017), stopword removal has been studied especially for the topic modelling task.

In the Word Mover's Distance paper, Kusner et al. (2015) used the stopword list provided by the SMART system (Salton, 1971), composed of 571 words. Stone et al. (2010) introduced another stopword list for English, with 339 words. Later, Puurula (2013) created a new stopword list with 988 words. Along this study, we found more than 10 different stopword lists that are currently being used across many machine learning and natural language processing tools.

5.4.2 *Cross Document-Topic Comparison*

Words that compose documents are sometimes not adequate to indicate the topics covered. Following this intuition, we augment the word space of each document by adding neighbors of each word. That way, the documents become more descriptive and carry more specific information. Our initial approach is to apply kNN search for each word in a document. Then, we either add these words' vectors inside the document features, or create a centroid of the word's neighbors and add it as a "topic-word".

Nevertheless, looking for the nearest neighbors of each word inside the global word vector space for every document is expensive. Thus, we apply clustering in the word vector space beforehand, and then search for the nearest neighbors of each word in the cluster that the word belongs to. Here we introduce the concept of "topic-words", which we refer to either neighbors or centroids of a word's neighbors. In our settings, we have used hard k-means clustering. After extracting these "topic-words", we cross compare with the Relaxed Word Mover's Distance approach. Finally, we quantify the distance as the mean of the previous two RWMD distances. An illustration of our proposed scheme is shown in Figure 5.5.

5.4.3 *Convex Metric Learning*

We extend our work in supervised settings, similarly to Supervised Word's Mover Distance (Huang et al., 2016). Here, we propose to replace the Neighborhood Component Analysis method, which includes a non-convex cost function (Goldberger et al., 2005) with the following convex ones.

LARGE MARGIN NEAREST NEIGHBORS The Large Margin Nearest Neighbors approach (LMNN) (Weinberger and Saul, 2009) is a convex loss metric that encourages inputs with similar labels to be close in a local region, and inputs of different labels to be farther by a large margin. LMNN is an algorithm to learn a Mahalanobis metric specifically to improve the classification error of k-nearest neighbors (kNN) classification (Cover and Hart, 1967).

More specifically, let $\{(\vec{x_i}, y_i)\}_{i=1}^{n}$ denote a training set of n labeled examples with inputs $\vec{x_i} \in \mathcal{R}^d$ and discrete (but not necessarily binary) class labels y_i. We use

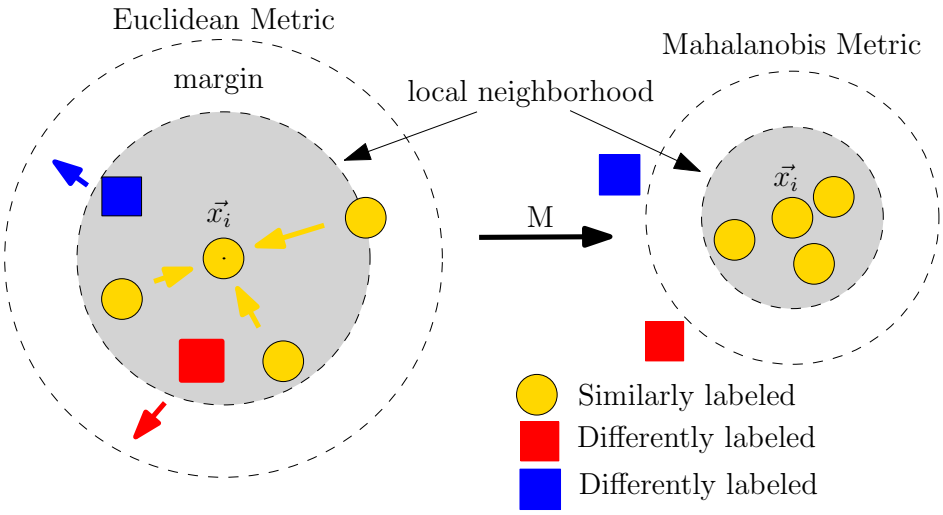


Figure 5.6: The LMNN algorithm for $k = 3$ target neighbors by Weinberger and Saul (2009).

the binary matrix $y_{ij} \in \{0, 1\}$ to indicate whether or not the labels y_i and y_j match. Our goal is to learn a linear transformation $L : \mathcal{R}^d \to \mathcal{R}^d$, which we will use to compute squared distances as:

$$D(x_i, x_j) = \|L(\vec{x_i} - \vec{x_j})\|^2 \tag{5.1}$$

Specifically, we want to learn the linear transformation that optimizes kNN classification when distances are measured in this way.

In addition to the class label y_i, for each input $\vec{x_i}$ we also specify k target neighbors

$$\epsilon(L) = \sum_{ij} \eta_{ij} \|L(\vec{x_i} - \vec{x_j})\|^2 + c \sum_{ijl} \eta_{ij}(1 - y_{il})[1 + \|L(\vec{x_i} - \vec{x_j})\|^2 - \|L(\vec{x_i} - \vec{x_l})\|^2]_+ \tag{5.2}$$

The second term in the cost function incorporates the idea of a margin. In particular, for each input $\vec{x_i}$, the hinge loss is incurred by differently labeled inputs whose distances do not exceed, by one absolute unit of distance, the distance from input $\vec{x_i}$ to any of its target neighbors. The cost function favors distance metrics in which differently labeled inputs maintain a large margin of distance and do not threaten to "invade" each other's neighborhoods. We show a schematic illustration of the LMNN approach in Figure 5.6 for an input with $k = 3$ target neighbors.

The optimization of eq. 5.2 can be reformulated as an instance of semidefinite programming (Vandenberghe and Boyd, 1996). A semidefinite program (SDP) is a linear program with the additional constraint that a matrix whose elements are linear in the unknown variables is required to be positive semidefinite. SDPs are convex; thus, with this reformulation, the global minimum of eq. 5.2 can be efficiently computed. To obtain the equivalent SDP, eq. 5.1 can be rewriten as:

$$D(\vec{x_i}, \vec{x_j}) = (\vec{x_i} - \vec{x_j})^\top M(\vec{x_i} - \vec{x_j}) \tag{5.3}$$

where the matrix $M = L^\top L$, parameterizes the Mahalanobis distance metric induced by the linear transformation L. Rewriting eq. 5.2 as an SDP in terms of M is straightforward, since the first term is already linear in $M = L^\top L$ and the hinge loss can be "mimicked" by introducing slack variables ξ_{ij} for all pairs of differently labeled inputs (i.e., for all h_i, j_i such that $y_{ij} = 0$). The resulting SDP is given by:

$$\begin{aligned} \min & \sum_{ij} \eta_{ij} (\vec{x_i} - \vec{x_j})^\top M(\vec{x_i} - \vec{x_j}) + c \sum_{ijl} \eta_{ij}(1 - y_{il})\xi_{ijl} \\ & \text{subject to:} \\ & (\vec{x_i} - \vec{x_j})^\top M(\vec{x_i} - \vec{x_j}) - (\vec{x_i} - \vec{x_j})^\top M(\vec{x_i} - \vec{x_j}) \geqslant 1 - \xi_{ijl} \\ & \xi_{ij} \geqslant 0 \\ & M \succeq 0 \end{aligned} \tag{5.4}$$

The last constraint $M \succeq 0$ indicates that the matrix M is required to be positive semidefinite.

MAXIMALLY COLLAPSING METRIC LEARNING Maximally Collapsing Metric Learning (MCML) (Globerson and Roweis, 2006) was introduced as a linear learning algorithm for quadratic Gaussian metrics (Mahalanobis distances) used in supervised classification tasks. The method is based on the simple geometric intuition that a good metric is one under which points in the same class are simultaneously near each other and far from points in the other classes. A convex optimization problem is formulated, whose solution generates such a metric by trying to collapse all instances within the same class to a single point, while pushing other class instances infinitely far away.

Given a set of n labeled examples (x_i, y_i), where $x_i \in \mathcal{R}^{\lceil}$ and $y_i \in \{1 \dots k\}$, we seek a similarity measure between two points in X space. We focus on Mahalanobis form metrics

$$d(x_i, x_j|A) = d_{ij}^A = (x_i - x_j)^\top A(x_i - x_j) \tag{5.5}$$

where A is a positive semidefinite (PSD) matrix. Intuitively, what we want from a good metric is that it makes elements of X in the same class look close whereas those in different classes appear far. Their approach starts with the ideal case when this is true in the most optimistic sense: same class points are at zero distance, and different class points are infinitely far. Alternatively this can be viewed as mapping x via a linear projection $Wx(A = WTW)$, such that all points in the same class are mapped into the same point. This intuition is related to the analysis of spectral clustering (Ng et al., 2002), where the ideal case analysis of the algorithm results in all same cluster points being mapped to a single point. To learn a metric which approximates the ideal geometric setup described above, for each training point, a conditional distribution over the points. For each x_i a conditional distribution over points $i \neq j$suchthat

$$p^A(j|i) = \frac{1}{Z_i} \exp^{-d_{ij}^A} = \frac{\exp^{-d_{ij}^A}}{\sum_{k \neq i} \exp^{-d_{ik}^A}} \qquad i \neq j \tag{5.6}$$

If all points in the same class were mapped to a single point and infinitely far from points in different classes, we would have the ideal "bi-level" distribution:

$$p_o(j|i) \propto \begin{cases} 1 & y_i = y_j \\ 0 & y_i \neq y_j \end{cases}$$

Furthermore, under very mild conditions, any set of points which achieves the above distribution must have the desired geometry. In particular, assume there are at least $\hat{d} + 2$ points in each class, where $\hat{d} = \text{rank}[A]$ (note that $\hat{d} \leqslant d$). Then $p^A(j|i) = p_o(j|i)(\forall i, j)$ implies that under A all points in the same class will be mapped to a single point, infinitely far from other class points.

Thus it is evident to find a matrix A such that $p^A(j|i)$ is as close as possible to $p_o(j|i)$. Then the KL divergence $KL[p_o|p]$ is minimized, as the objective is to match distributions:

$$\min_A \sum_i KL[p_o(j|i)|p^A(j|i)] \quad \text{s.t.} A \in \text{PSD} \tag{5.7}$$

Dataset	n	Voc	Avg	|y|
BBCSPORT	517	13,243	117	5
TWITTER	2,176	6,344	9.9	3
RECIPE	3,059	5,708	48.5	15
OHSUMED	3,999	31,789	59.2	10
CLASSIC	4,965	24,277	38.6	4
REUTERS	5,485	22,425	37.1	8
AMAZON	5600	42063	45.0	4
20NEWS	11293	29671	72	20

Table 5.1: Dataset statistics.

The authors use a first order gradient method, specifically the projected gradient approach. At each iteration they take a small step in the direction of the negative gradient of the objective function, followed by a projection back onto the PSD cone. This projection is performed simply by taking the eigen-decomposition of A and removing the components with negative eigenvalues.

Both methods carry the property of learning a metric where points in the same class are simultaneously near each other and far from points in the other classes.

5.4.4 *Experiments*

We evaluate the proposed methods in the context of kNN classification on six benchmark document categorization tasks:

1. BBCSPORT: BBC sports articles between 2004- 2005,
2. TWITTER: a set of tweets labeled with sentiments 'positive', 'negative', or 'neutral' (Sanders, 2011),
3. RECIPE: a set of recipe procedure descriptions labeled by their region of origin,
4. OHSUMED: a collection of medical abstracts categorized by different cardiovascular disease groups (for computational efficiency we subsample the dataset, using the first 10 classes),
5. CLASSIC: sets of sentences from academic papers, labeled by publisher name,

6. REUTERS: a classic news dataset labeled by news topics (we use the 8-class version with train/test split as described in Cachopo (2007)),

In Table 5.1 we present some statistics for the used datasets.

EXPERIMENTAL SETUP For comparison purposes, we use the train/test splits provided by previous work (Kusner et al., 2015). Datasets are pre-processed by removing all words in the SMART stopword list (Salton, 1971), except TWITTER. We make use of the pre-trained version of word embeddings, offering more than three million words/phrases (from Google News), trained using the skip-gram variant of word2vec, (Mikolov et al., 2013b). Words that do not exist in the word2vec pre-trained model vocabulary, are removed. For the stopword removal method, on top of the SMART list removal, we use another stopword list (Stone et al., 2010), consisting of 339 words, which is used in popular libraries like Gensim* and spaCy†. In k-means clustering we tried a range of 50, 100, 200, 500 clusters, with 500 giving best results..

RESULTS We evaluate our approaches against WMD (Kusner et al., 2015), LSI (Deerwester et al., 1990), and Supervised WMD (Huang et al., 2016). The effectiveness of the learned metrics is assessed by the kNN classification error.

Table 5.2 demonstrates results of our proposed "bag-of-tricks" to boost Word Mover's Distance. Our unsupervised approaches achieve state-of-the-art results in four out of six datasets. Stopword removal with various resources can assist the embeddings and reach the Supervised WMD accuracy in the case of the TWITTER dataset. As expected, removing unnecessary stopwords can help WMD dramatically, especially in small size documents.

Next, adding neighbors of words that exist in a document, can further enhance the "topical" expression and thus result in better distance computation. We observed that, by incrementing a document with words that are close in the word embedding space, we achieve better accuracy than traditional WMD or LSI approach in most cases. Utilizing prior clustering in word vectors can further boost neighbor words that belong in semantically closer clusters or groups, especially in very small or very large document sizes.

* https://radimrehurek.com/gensim/
† https://spacy.io/

		BBCSPORT	TWITTER	RECIPE	OHSUMED	CLASSIC	REUTERS
Unsupervised	LSI	4.30 ± 0.60	31.70 ± 0.70	45.40 ± 0.50	44.20	6.70 ± 0.40	6.30
	WMD	4.60 ± 0.70	28.70 ± 0.60	42.60 ± 0.30	44.50	2.88 ± 0.10	3.50
	Stop RWMD	4.27 ±1.19	27.51 ± 1.00	43.98 ± 1.40	44.27	3.25 ± 0.50	5.25
	All, 5nn	6.00±1.34	29.23 ± 1.09	42.52 ± 1.18	46.73	3.18 ± 0.44	6.26
	All, 5nn, Mean	4.00 ± 1.55	28.58 ± 2.29	42.53 ± 0.67	43.90	3.08 ± 0.62	5.76
	k-m, 5nn	5.91 ± 2.65	28.56 ± 1.20	42.23 ± 1.15	46.50	2.98 ± 0.66	4.71
	k-m, 5nn, Mean	3.82 ± 1.72	28.50 ± 1.51	41.95 ± 1.04	44.05	3.08 ± 0.51	4.57
Supervised	S-WMD (NCA)	2.10 ± 0.50	27.50 ± 0.50	39.20 ± 0.30	**34.30**	3.20 ± 0.20	3.20
	LMNN	**1.73 ± 0.67**	28.86 ± 2.22	40.88 ± 1.88	39.59	**2.76 ± 0.30**	4.02
	MCML	2.45 ± 1.27	**27.15 ± 1.36**	**38.93 ± 1.24**	42.38	3.56 ± 0.49	**2.92**

Table 5.2: Comparison in kNN test error(%) to LSI, WMD and S-WMD. Blue shows best results in unsupervised methods and bold indicates best result for a dataset.

Finally, suggesting LMNN and MCML, as convex minimization algorithms for initializing Supervised WMD, provided the best results across five datasets, due to their convex nature. While MCML achieved very good performance, its training required a notable amount of time, compared to LMNN. In particular, our goal was to signify the potential of alternative metric learning methods as one of the most intriguing components of Supervised WMD. The final results show that the supervised version of WMD requires further investigation, since a small change in the initialization part affected the computation.

5.5 NEURAL NETWORKS FOR LEARNING SET REPRESENTATIONS

In a variety of domains, complex data objects can be expressed as compositions of other, simpler objects. These simpler objects naturally correspond to the parts or components of the complex objects. For instance, in natural language processing, documents may be represented by sets of word embeddings. Likewise, in graph mining, a graph may be viewed as a set of vectors where these vectors correspond to the embeddings of its nodes. In computer vision, images may be described by local features extracted from different regions of the image. In such scenarios, one set of feature vectors denotes a single instance of a particular class of interest (an object, document, graph, etc.). Performing machine learning tasks on such types of objects (e.g., set classification, set regression, etc.) is very challenging. While typical machine learning algorithms are designed for fixed dimensional data

instances, the cardinalities of these sets are not fixed, but they are allowed to vary. Furthermore, the components of the sets usually do not have an inherent ordering. Hence, machine learning algorithms have to be invariant to permutations of these components.

Traditionally, the most common approach to the problem is to define a distance/similarity measure or kernel that finds a correspondence between a pair of sets, and to combine it with an instance-based machine learning algorithm such as kNN or SVM. This approach has dominated the field, and has achieved state-of-the-art results on many datasets. However, its main disadvantage is that it is a two-step approach. Data representation and learning are independent from each other. Ideally, we would like to have an end-to-end approach. Besides the above problem, these methods usually suffer from high computational and memory complexity since they compare all sets to each other.

In recent years, neural network architectures have proven extremely successful on a wide variety of tasks, notably in computer vision, in natural language processing, and in graph mining. One of the main reasons of the success of neural networks is that the representation of data is adapted to the task at hand. Specifically, recent neural network models are end-to-end trainable, and they generate features that are suitable for the task at hand. Most models that operate on sets usually update the representations of the components of the set using some architectures, typically a stack of fully connected layers. And then, they apply a permutation invariant function to the updated component representations to obtain a representation for the set. Although these networks have proven successful in several tasks, they usually employ very simple permutation invariant functions such as the sum or the average of the representations of the components. This may potentially limit the expressive power of these architectures.

In this part, we propose a novel neural network architecture for performing machine learning tasks on sets of vectors. The network is capable of generating representations for unordered, variable-sized feature sets. Interestingly, the proposed model produces exactly the same output for all possible permutations of a set of vectors.

To achieve that, we generate a number of hidden sets and we compare the input set with these sets using a network flow algorithm such as bipartite matching (BM). The outputs of the network flow algorithm form the penultimate layer and

are passed to a fully-connected layer which produces the output. Since the employed network flow algorithms are differentiable, we can update these hidden sets during training with backpropagation. Hence, the proposed neural network is end-to-end trainable, while the hidden sets are different for each problem considered. Deeper models can be obtained by stacking more fully-connected layers one after another. We demonstrate the proposed architecture in a variety of classification tasks: object recognition from sets of points that define the shape of objects, text categorization from sets of word embeddings, and graph classification from bags of node embeddings. The results show that the proposed model produces better or competitive results with state-of-the-art techniques, while its time complexity remains attractive.

Our main contributions are summarized as follows:

- We propose a novel architecture, RepSet, for performing machine learning on sets which, in contrast to traditional approaches, is capable of adapting data representation to the task at hand.
- We also propose a relaxed version of the previous method, called Approx-RepSet, for large data volumes, higher dimensions and bigger document sizes.
- We evaluate the proposed architecture on several benchmark datasets, and achieve state-of-the-art performance.

5.5.1 *Methodology*

Conventional machine learning algorithms are designed to operate on fixed-size feature vectors, and they are thus unable to handle sets. In our setting, each example is represented as a collection $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ of d-dimensional vectors, $\mathbf{v}_i \in \mathbb{R}^d$. Note that examples are allowed to vary in the number of elements. Hence, it is not necessary that all examples comprise of exactly n components. Since our input is a set $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, $v_i \in \mathbb{R}^d$, our input domain is the power set $\mathcal{X} = 2^{\mathbb{R}^d}$, and we would like to design an architecture whose output would be the same regardless of the ordering of the elements of X. Clearly, to achieve that, a permutation invariant function is necessary to be introduced at some layer of the architecture. In fact, the model that we propose consists sim-

ply of standard fully-connected layers along with a permutation invariant layer. Hence, we next present that permutation invariant layer in detail.

In this part, we propose a novel permutation invariant layer which capitalizes on well-established concepts from flows and matchings in graphs. The proposed layer contains m hidden sets $Y_1, Y_2, \ldots, Y_m$ of d-dimensional vectors. These sets may have different cardinalities and their components are trainable, i. e., the elements of a hidden set Y_i correspond to the rows of a trainable matrix $\mathbf{W}_i$. Therefore, each column of matrix $\mathbf{W}_i$ is a vector $\mathbf{u} \in Y_i$.

A natural way to measure the similarity between the input set and each one of the hidden sets is by comparing their building blocks, i. e., their components. To achieve that, we capitalize on network flow algorithms. Specifically, we use the bipartite matching algorithm to compute a correspondence between the elements of the input set X and the elements of each hidden set Y_i.

In our setting, the input set X corresponds to set V_1, the hidden set Y_i corresponds to set V_2, and the bipartite graph is fully connected, i. e., every element of V_1 is connected to all the elements of V_2. The weight of each edge is the result of a differentiable function f on the representations of the edge's two endpoints.

Formally, given an input set of vectors, $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k_1}\}$ and a hidden set $Y_i = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{k_2}\}$, we can obtain the maximum matching between the elements of the two sets by solving the following linear program:

$$
\begin{aligned}
&\max \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} x_{ij} f(\mathbf{v}_i, \mathbf{u}_j) \\
&\text{subject to} \\
&\sum_{i=1}^{k_1} x_{ij} \leqslant 1 \quad \forall j \in \{1, \ldots, k_2\} \\
&\sum_{j=1}^{k_2} x_{ij} \leqslant 1 \quad \forall i \in \{1, \ldots, k_1\} \\
&x_{ij} \geqslant 0 \quad \forall i \in \{1, \ldots, k_1\}, \forall j \in \{1, \ldots, k_2\}
\end{aligned}
\tag{5.8}
$$

where $f(\mathbf{v}_i, \mathbf{u}_j)$ is, as mentioned above, a differentiable function, and $x_{ij} = 1$ if component i of X is assigned to component j of Y_i and 0 otherwise. In our experiments, we have defined $f(\mathbf{v}_i, \mathbf{u}_j)$ as the inner product between the two vectors $\mathbf{v}_i$,

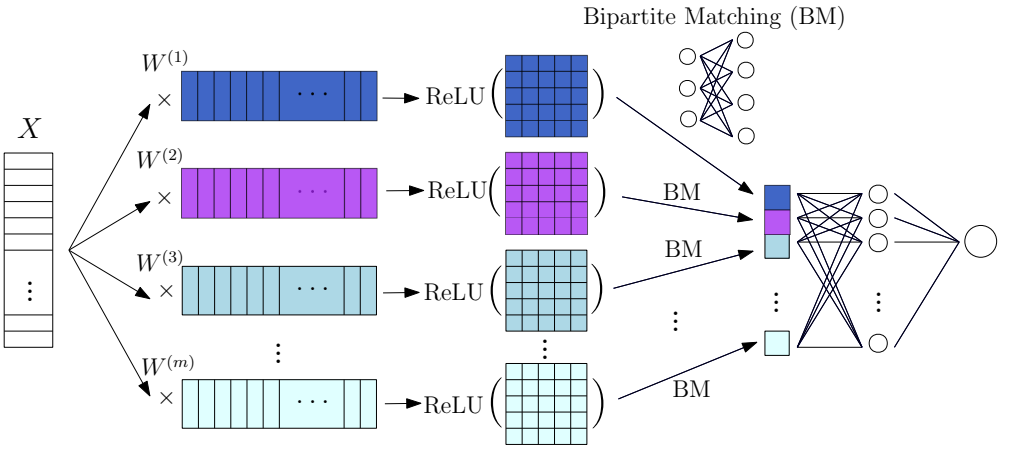


Figure 5.7: Our RepSet architecture.

and $\mathbf{u}_j$ followed by the ReLU activation function. Hence, $f(v_i, u_j) = \text{ReLU}(\mathbf{v}_i^\top \mathbf{u}_j)$. We further define the matrix $\mathbf{G}^{(i)}$ for each set Y_i such as $\mathbf{G}^{(i)}_{jk} = f(v_j, v_k)$.

We also define the matrix $D^{(i)} \in \mathbb{R}^{k_1 \times k_2}$ such as $D^{(i)}_{jk} = x_{jk}$. Given an input set X and the m hidden sets $Y_1, Y_2, \ldots, Y_m$, we formulate m different bipartite matching (BM) problems, and by solving all m of them, we end up with an m-dimensional vector $\mathbf{v}_X$ which corresponds to the hidden representation of set X. This m-dimensional vector can be used as features for different machine learning tasks such as set regression or set classification. For instance, in the case of a set classification with c classes, the output is computed as follows:

$$\mathbf{p}_X = \text{softmax}(\mathbf{W}^{(c)}\mathbf{v}_X + \mathbf{b}^{(c)}) \tag{5.9}$$

With $\mathbf{W}^{(c)} \in \mathbb{R}^{m \times c}$ trainable matrix and $b^{(c)} \in \mathbb{R}^c$ being the bias term. We use the negative log likelihood of the correct labels as training loss:

$$L = -\sum_{c_i \in c} \delta_{c_X}(c_i) \log \mathbf{p}_{X_{c_i}} \tag{5.10}$$

where i is the class label of set X. Note that we can create a deeper architecture by adding more fully-connected layers. In Figure 5.7 we present an illustration of our proposed architecture.

As mentioned above, a constraint that the neural network is necessary to satisfy is to be invariant under any permutation of the elements of the input set. Our

next theoretical result shows that the proposed model generates the same output for all $n!$ permutations of the elements of X.

Theorem 5.1. *Let X be a set having elements from a countable or uncountable universe. The proposed architecture is invariant to the permutation of elements in X.*

Proof. We set Π_n be the set of all permutations of the integers from 1 to n. Let's pick $\pi \in \Pi_{k_1}$ and apply it to the set X. The bipartite matching problem then becomes :

$$\begin{aligned}
&\max \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} x_{ij} f(\mathbf{v}_{\pi(i)}, \mathbf{u}_j) \\
&\text{subject to} \\
&\sum_{i=1}^{k_1} x_{ij} \leqslant 1 \quad \forall j \in \{1, \dots, k_2\} \\
&\sum_{j=1}^{k_2} x_{ij} \leqslant 1 \quad \forall i \in \{1, \dots, k_1\} \\
&x_{ij} \geqslant 0 \quad \forall i \in \{1, \dots, k_1\}, \forall j \in \{1, \dots, k_2\}
\end{aligned} \tag{5.11}$$

since the sum of the elements of a whole set is permutation invariant it ensures that for every permutations in Π_{k_1} applied to x_{ij} the constraints will remain the same.
Moreover:

$$\sum_{i=1}^{k_1} \sum_{j=1}^{k_2} x_{ij} f(\mathbf{v}_{\pi(i)}, \mathbf{u}_j) = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} x_{\pi^{-1}(i)j} f(\mathbf{v}_i, \mathbf{u}_j) \tag{5.12}$$

the set giving the optimal solution of problem (5.8), say $\{x^*_{\pi^{-1}(i)j}, \forall i \in \{1, \dots, k_1\}, \forall j \in \{1, \dots, k_2\}\}$ provides the same maximal solution as problem (5.11) modulo one the permutation π. As those arguments providing the same maximum for both problems it ensures the permutation invariance of the bipartite matching. □

Let's now focus on the gradient of the loss:

$$\frac{\partial L}{\partial \mathbf{W}_j^{(c)}} = \mathbf{p}_j - \mathbf{y}_j \tag{5.13}$$

where $\mathbf{W}_j^{(c)}$ is the j^{th} row of $\mathbf{W}^{(c)}$. Before going any further, we also need to define the following differentiable function:

$$g(\mathbf{W}^{(k)}) = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} \mathbf{D}_{ij}^{(k)*} f(\mathbf{v}_i, \mathbf{u}_j^{(k)}) \tag{5.14}$$

where $\mathbf{D}^{(k)*}$ is the arg max of the optimisation problem (5.8) for the corresponding Y_k.

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{W}^{(k)}} &= \frac{\partial L}{\partial \mathbf{W}_k^{(c)}} \frac{\partial \mathbf{W}_k^{(c)}}{\partial \mathbf{W}^{(k)}} = (\mathbf{p}_k - \mathbf{y}_k) \frac{\partial \mathbf{W}_k^{(c)}}{\partial \mathbf{W}^{(k)}} \\
&= (\mathbf{p}_k - \mathbf{y}_k) \frac{\partial}{\partial \mathbf{W}^{(k)}} \sigma(g(\mathbf{W}^{(k)})) \\
&= (\mathbf{p}_k - \mathbf{y}_k) \sigma(g(\mathbf{W}^{(k)}))(1 - \sigma(g(\mathbf{W}^{(k)}))) \\
&\quad \frac{\partial}{\partial \mathbf{W}^{(k)}} g(\mathbf{W}^{(k)}).
\end{aligned}$$

then

$$\frac{\partial}{\partial \mathbf{W}^{(k)}} g(\mathbf{W}^{(k)}) = \frac{\partial}{\partial \mathbf{W}^{(k)}} \mathrm{tr}(\mathbf{D}^{(k)*\top} \mathbf{G}^{(k)}) \tag{5.15}$$

We consider as $\mathbf{X}$ the matrix representation of set X. Since we have $\mathbf{G}^{(k)} = \mathrm{ReLU}(\mathbf{XW})$ the points that have been nullified by the ReLU function during the forward pass are stored as zero values in the optimal solution $\mathbf{D}^{(k)*}$. Indeed, no edge was created whenever the value of the dot product was negative. Then none of these pairs have been used during the forward pass. This yields :

$$\frac{\partial}{\partial \mathbf{W}^{(k)}} g(\mathbf{W}^{(k)}) = \frac{\partial}{\partial \mathbf{W}^{(k)}} \mathrm{tr}(\mathbf{D}^{(k)*\top} \mathbf{X} \mathbf{W}^{(k)}) \tag{5.16}$$

$$\frac{\partial}{\partial \mathbf{W}^{(k)}} g(\mathbf{W}^{(k)}) = \mathbf{X}^\top \mathbf{D}^{(k)*} \tag{5.17}$$

Which finally gives:

$$\frac{\partial L}{\partial \mathbf{W}^{(k)}} = (\mathbf{p}_k - \mathbf{y}_k) \sigma(g(\mathbf{W}^{(k)}))(1 - \sigma(g(\mathbf{W}^{(k)}))) \mathbf{X}^\top \mathbf{D}^{(k)*} \tag{5.18}$$

The major weakness of the above architecture is its computational complexity. Computing a maximum cardinality matching in a weighted bipartite graph with n vertices and m edges takes time $\mathcal{O}(mn + n^2 \log n)$, using the classical Hungarian

algorithm. This prohibits the proposed model from being applied to very large datasets. To account for that, we next present an approximation of the bipartite matching problem which involves operations that can be performed on a GPU, allowing thus efficient implementations.

More specifically, given an input set of vectors, $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{k_1}\}$ and a hidden set $Y_i = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{k_2}\}$, first we identify which of the two sets has the highest cardinality. If $|X| \geqslant |Y_i|$, we solve the following linear program:

$$\begin{aligned} &\max \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} x_{ij} f(\mathbf{v}_i, \mathbf{u}_j) \\ &\text{subject to} \\ &\sum_{i=1}^{k_1} x_{ij} \leqslant 1 \quad \forall j \in \{1, \ldots, k_2\} \\ &x_{ij} \geqslant 0 \quad \forall i \in \{1, \ldots, k_1\}, \forall j \in \{1, \ldots, k_2\} \end{aligned} \tag{5.19}$$

Conversely, if $|X| < |Y_i|$, then we replace the first constraint with the following $\sum_{j=1}^{k_2} x_{ij} \leqslant 1 \quad \forall i \in \{1, \ldots, k_1\}$. This relaxed problem must yield an upper-bound to the Bipartite Matching problem as any solution satisfying the it must remain feasible if one constraint is removed.

Proposition 5.1. *The optimization problem defined in Equation 5.19 is an upper bound to the bipartite matching problem defined in Equation 5.8*

Proof. We define an optimal solution to the relaxed problem, namely $\mathbf{D}^*$, such as:

$$\mathbf{D}^*_{ij} = \begin{cases} 1 & \text{if } i = \arg\max_i f(v_i, u_j) \\ 0 & \text{otherwise} \end{cases}$$

The relaxed problem actually enables us to pick an edge of the bipartite graph with weight $f(v_i, u_j)$, even if we already picked another one with sink u_j. The optimality of this solution then comes from the following fact. We always pick the maximal weight for every pair of nodes, regardless if it was already picked from another one on the restricted side of the bipartite graph. Then, with $i^* =$

$\arg\max_i f(v_i, u_j)$, the optimality of this solution follows for non relaxed problem. As with any $\mathbf{D}$ feasible solution subject to both constraints, for any j we have:

$$\begin{aligned}\sum_{i=1}^{k_1} \mathbf{D}_{ij} f(\mathbf{v}_i, \mathbf{u}_j) &\leqslant \sum_{i=1}^{k_1} \mathbf{D}_{ij} f(\mathbf{v}_i, \mathbf{u}_j *) \\ &= f(\mathbf{v}_i, \mathbf{u}_j *) \sum_{i=1}^{k_1} \mathbf{D}_{ij} \\ &\leqslant f(\mathbf{v}_i, \mathbf{u}_j *) = \sum_{i=1}^{k_1} \mathbf{D}^*_{ij} f(\mathbf{v}_i, \mathbf{u}_j)\end{aligned} \tag{5.20}$$

Therefore, $\mathbf{D}^*$ must yield a maximal objective value for the bipartite matching problem.

□

5.5.2 *Experiments*

DATASETS As previously, we evaluate the proposed methods in the context of kNN classification on six benchmark document categorization tasks: (1) BBCSPORT: BBC sports articles between 2004-2005; (2) TWITTER: a set of tweets labeled with sentiments 'positive', 'negative', or 'neutral'; (3) RECIPE: a set of recipe procedure descriptions labeled by their region of origin; (4) OHSUMED: a collection of medical abstracts; (5) CLASSIC: sets of sentences from academic papers, labeled by publisher name; (6) REUTERS: a classic news dataset labeled by news topics (Cachopo, 2007); (7) AMAZON: a set of Amazon reviews which are labeled by category product in books, dvd, electronics, kitchen (as opposed to by sentiment); (8) 20NG: news articles classified into 20 different categories (we use the "bydate" train/test split by Cachopo (2007)). We refer to Table 5.1 for datasets' statistics.

EXPERIMENTAL SETUP We preprocess all datasets by removing all words in the SMART stopword list (Salton, 1971) like the configuration of Kusner et al. (2015). For the size of latent sets as well as their document size, we choose between $\{10, 20, 30, 50, 100\}$. We also examine different batch sizes ranging in $\{8, 16, 32, 64\}$. For optimization, the learning rate was set to 0.01.

	BBCSPORT	TWITTER	RECIPE	OHSUMED	CLASSIC	REUTERS	AMAZON	20NG
LSI	4.30 ± 0.60	31.70 ± 0.70	45.40 ± 0.50	44.20	6.70 ± 0.40	6.30	9.30 ± 0.40	28.90
WMD	4.60 ± 0.70	28.70 ± 0.60	42.60 ± 0.30	44.50	**2.88** ± 0.10	3.50	7.40 ± 0.30	26.80
S-WMD	2.10 ± 0.50	27.50 ± 0.50	39.20 ± 0.30	34.30	3.20 ± 0.20	3.20	5.80 ± 0.10	26.80
DeepSets	25.45 ± 20.1	29.66 ± 1.62	70.25	71.53	5.95 ± 1.50	10.00	8.58 ± 0.67	38.88
NN-mean	10.09 ± 2.62	31.56 ± 1.53	64.30 ± 7.30	45.37	5.35 ± 0.75	11.37	13.66 ± 3.16	38.40
NN-max	2.18 ± 1.75	30.27 ± 1.26	43.47 ± 1.05	35.88	4.21 ± 0.11	4.33	7.55 ± 0.63	32.15
NN-attention	4.72 ± 0.97	29.09 ± 0.62	43.18 ± 1.22	**31.36**	4.42 ± 0.73	3.97	6.92 ± 0.51	28.73
RepSet	**2.00** ± 0.89	**25.42** ± 1.10	**38.57** ± 0.83	**33.88**	3.35 ± 0.32	3.15	**5.29** ± 0.28	**22.98**
Approx	4.27 ± 1.73	27.40 ± 1.95	40.94 ± 0.40	35.94	3.76 ± 0.45	**2.83**	5.69 ± 0.40	23.82

Table 5.3: Comparison in kNN test error(%). Bold shows best results.

RESULTS We observe that RepSet achieves comparable to state-of-the-art techniques like WMD, LSI, Supervised WMD, DeepSets and attention models. This is because of its power to learn the hidden representations based on the distances computed by the weighted bipartite matching. RepSet performs better in 7 out of 8 text classification datasets. Surprisingly, the approximate algorithm, ApproxRepSet, performed better than the exact RepSet for the REUTERS dataset. This is because ApproxRepSet is significantly faster than the exact version and thus with small parameter tuning we had better results. Since the computation of the exact RepSet in Matlab was slow, we did not have the time to perform extensive parameter tuning.

It is remarkable that our RepSet model performed better than the state-of-the-art distance-based Supervised Word Mover's Distance, which involves complex metric learning and exhaustive parameter tuning. In 4 datasets, the Supervised WMD is surpassed even by our approximate RepSet, with a significant smaller complexity.

Table 5.3 shows the average classification error of the proposed models and those of the baselines. On all datasets except two (OHSUMED, CLASSIC), the proposed model outperforms the baselines. In some cases, the gains in accuracy over the best performing competitors are considerable. For instance, on the 20NG, TWITTER, and RECIPE datasets, RepSet achieves respective absolute improvements of 3.82%, 2.08% and 0.63% in accuracy over the best competitor, the S-WMD method. Furthermore, the proposed model outperforms DeepSets on all datasets, and on most of them by wide margins. Overall, it is clear from Table **??** that RepSet is superior to the other methods in the text categorization task. Regard-

Hidden set	Terms similar to elements of hidden sets	Terms similar to centroids of hidden sets
1	chelsea, football, striker, club, champions	footballing
2	qualify, madrid, arsenal, striker, united, france	ARSENAL_Wenger
3	olympic, athlete, olympics, sport, pentathlon	Olympic_Medalist
4	penalty, cup, rugby, coach, goal	rugby
5	match, playing, batsman, batting, striker	batsman

Table 5.4: Terms of the employed pre-trained model that are most similar to the elements and centroids of elements of 5 hidden sets.

ing the two variants of the proposed architecture, the model that solves exactly the bipartite matching problem (RepSet) outperforms the model that approximates it (ApproxRepSet) on all datasets except from REUTERS. However, on most datasets the difference in performance is not large. Hence, although less powerful, the approximate model is still capable of learning expressive representations of sets, and it can thus provide the means for tackling computationally challenging tasks.

We should mention at this point that besides effective, the proposed model is also highly interpretable. For instance, in the text categorization setting, the elements of each hidden set can be regarded as the terms of hidden documents which are likely to be related to the topics of the different classes. To experimentally verify that, we trained a model with 50 hidden sets on the BBCSPORT dataset. Each hidden set consisted of 20 vectors. We found the terms that are closer to these vectors and then computed the centroid for 5 of the hidden sets. Table 5.4 shows the terms of the pre-trained model that were found to be most similar (using cosine similarity), with respect to the vectors and centroids. Clearly, the centroids of these 5 hidden sets are close to terms that are related to sports. Interestingly, some of these terms correspond to cricket positions, as well as to names of famous soccer players.

Apart from text classification, we tested our approach also on the graph classification task, getting also state-of-the-art results. As this application falls out of the scope of this Ph.D. thesis, we point the reader to our paper.

RUNTIME ANALYSIS To evaluate the runtime performance and scalability of the proposed models, we created a series of synthetic datasets and measured how the average running time per epoch varies with respect to the parameters of the

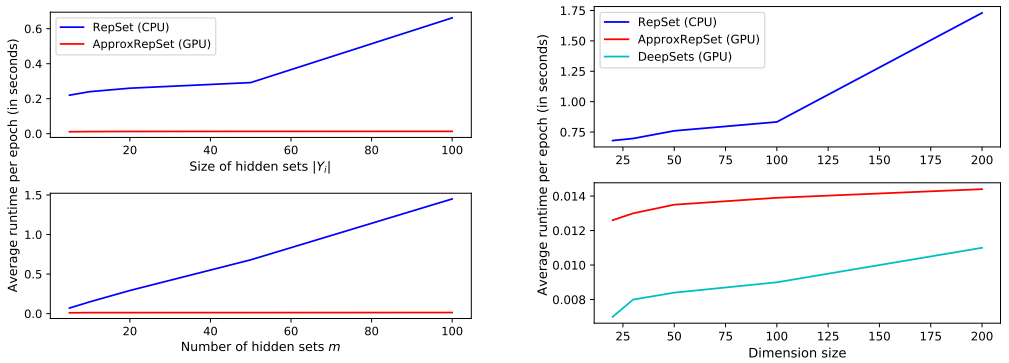


Figure 5.8: Runtimes with respect to the number of hidden sets m, the size of the hidden sets $|Y_i|$ (left) and embeddings with different dimensions (right).

model. Figure 5.8 illustrates the running time of RepSet and ApproxRepSet as a function of the size of the hidden sets $|Y_i|, i = 1, \ldots, m$ (top left) and as a function of the number of hidden sets m (bottom left). Note that for RepSet, training was performed on an Intel Xeon $E5-1607$ CPU (4 threads), while for ApproxRepSet, it was performed on an NVIDIA Titan Xp GPU. As expected, we observe that RepSet is more computationally expensive than ApproxRepSet, while its running time increases significantly as the size and the number of hidden sets increase. On the other hand, the values of these parameters do not have a large impact on the running time of ApproxRepSet. We also evaluate (Figure 5.8 (right)) how the proposed models scale as the dimensionality of the vectors contained in the sets increases and compare them against the DeepSets model. DeepSets is the fastest model, followed by ApproxRepSet. The running times of both these models are much smaller than that of RepSet, while they also grow very slowly as the dimensionality of vectors increases. Conversely, the running time of RepSet increases notably especially for dimensionalities larger than 100. We also examine in Figure 5.9 how the running time of the three models varies with respect to the number of input samples N (left) and to their cardinality $|X_i|, i = 1, \ldots, N$ (right). Surprisingly, we find that the running time of RepSet grows slowly as the number of samples increases. On the other hand, it increases significantly as the the cardinality of these samples increases. The running times of DeepSets and ApproxRepSet are again much lower than that of RepSet, and grow only slightly as the number of samples and their cardinality increase.

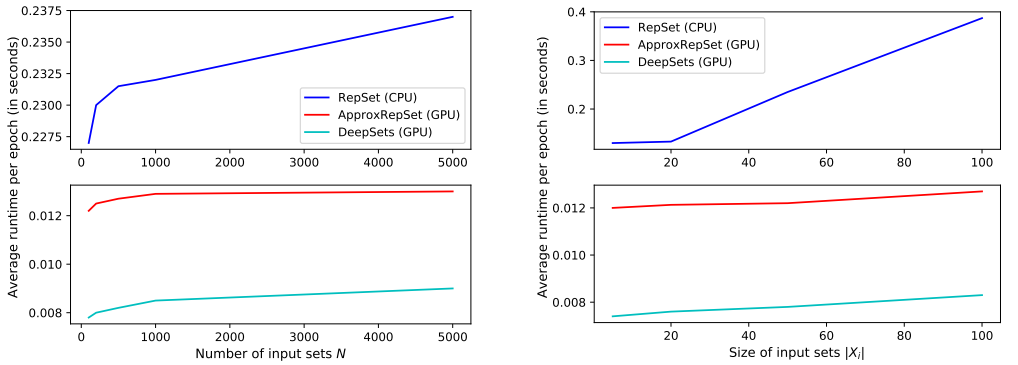


Figure 5.9: Runtimes with respect to the number of input sets N (left) and the size of the input sets $|X_i|$ (right).

SENSITIVITY ANALYSIS The proposed RepSet and ApproxRepSet models involve two main parameters: (1) the number of hidden sets m, and (2) the cardinalities of the hidden sets $|Y_i|, i = 1, \ldots, m$. We next investigate how these two parameters influence the performance of the RepSet model. Specifically, in Figures 5.10 and 5.11, we examine how the different choices of these parameters affect the performance of RepSet on the TWITTER and RECIPE datasets, respectively. We measure the test error as a function of the two parameters. Note that each hidden set Y_i can have a different cardinality compared to the other sets. However, we set the cardinalities of all hidden sets to the same value. We observe that on TWITTER, the number of hidden sets m does not have a large impact on the performance, especially for small cardinalities of the hidden sets ($|Y_i| \leqslant 50$). For most cardinalities, the test error is within 1% to 3% when varying this parameter. Furthermore, in most cases, the best performance is attained when the number of hidden sets is small ($m \leqslant 20$). Similar behavior is also observed for the second parameter on the TWITTER dataset. For most values of m, the test error changes only slightly when varying the cardinalities of the hidden sets. For $m \geqslant 50$, the model produces best results when the cardinalities of the hidden sets $|Y_i|$ are close to 20. On the other hand, for small values of m, the model yields good performance even when the cardinalities of the hidden sets $|Y_i|$ are large. On the RECIPE dataset, both parameters have a higher impact on the performance of the RepSet model. In general, small values of m lead to higher test error than larger values of m. For most values of $|Y_i|$, values of m between 20 and 50 result in the lowest test error. As regards the size of the hidden sets $|Y_i|$, there is no consistency in the obtained results. Specifically, for small values of m ($m \leqslant 20$), large cardinalities of the hidden sets result

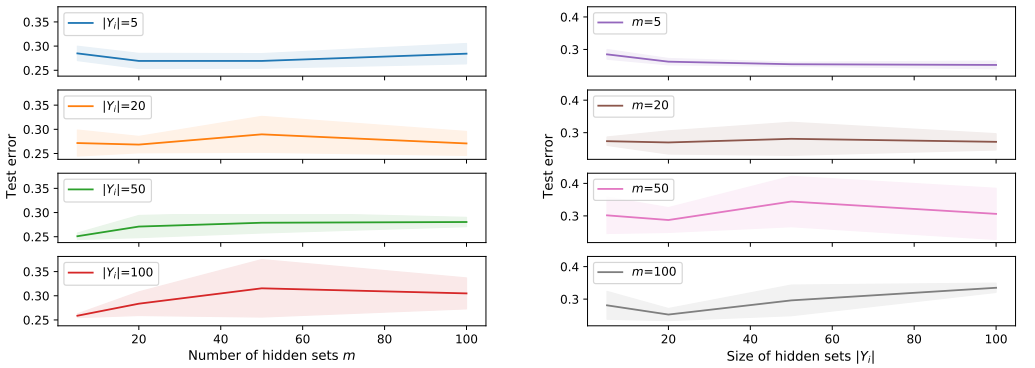


Figure 5.10: Average test error of RepSet with respect to the number of hidden sets m (left) and the size of the hidden sets $|Y_i|$ (right) on the TWITTER dataset.

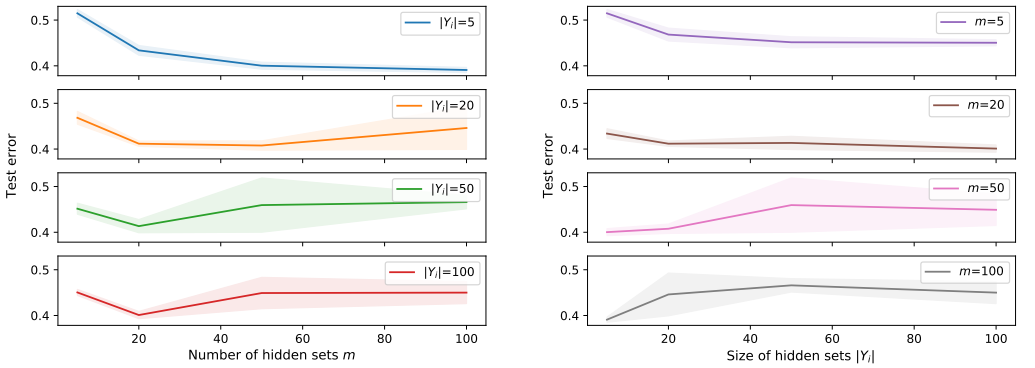


Figure 5.11: Average test error of RepSet with respect to the number of hidden sets m (left) and the size of the hidden sets $|Y_i|$ (right) on the RECIPE dataset.

in better performace, while for large values of m ($m \geqslant 50$), small cardinalities lead to smaller error.

5.6 CONCLUSION & FUTURE WORK

SUMMARY In this part of the thesis, we presented effective and efficient boosting tricks for improving Word Mover's Distance speed and accuracy. We empirically pointed out a number of possible adjustments for the existing WMD, such as stopword removal, cross document-topic comparison and convex metric learning methods. Calibrating those three components (two unsupervised and one

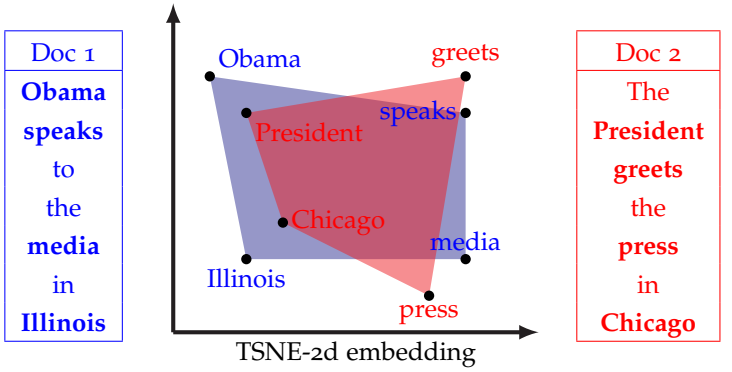


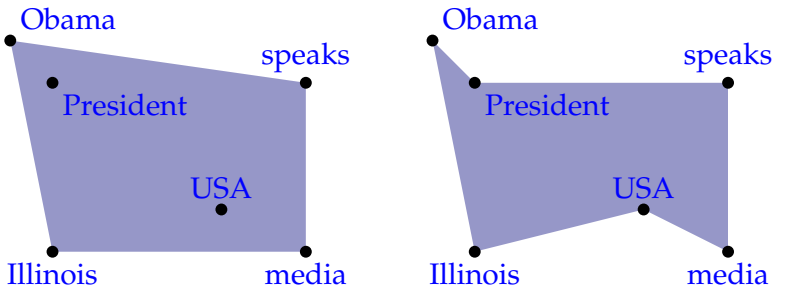
Figure 5.12: Areas covered by two documents in the word vector space.

supervised), we managed to achieve lower error in the task of text categorization compared to the original WMD and its supervised counterpart.

FUTURE WORK Measuring similarity between two documents that share words appearing in different context, can make the comparison process harder. Thus, the problem of polysemy should also be addressed. In order to address the phenomenon of words appearing in multiple contexts, Topical Word Embeddings (Liu et al., 2015) can be applied. Thus, a "topical" WMD, based on topics rather than documents alone, would be a promising direction step. Recently new techniques for learning word representations have been proposed, like the Poincaré (Nickel and Kiela, 2017), ELMo (Peters et al., 2018) and BERT embeddings (Devlin et al., 2018), as they could improve WMD dramatically. Finally, we plan to fully examine non-Linear Metric Learning (Kedem et al., 2012) methods for the supervised version of WMD. Gradient Boosting LMNN or χ^2-LMNN could provide better results with the use of their non-linear nature.

In future work we want to examine new metrics for measuring distance between text documents, using methodologies from computational geometry. Our idea, is built upon the work of Kusner et al. (2015), combining the powerful word embeddings Mikolov et al. (2013b) with traditional approaches for shape comparisons.

We represent text documents as convex hull polygons in the word embeddings space. The space could be already dropped with dimensionality techniques like Maaten and Hinton (2008). The distance between two text documents A and B is given by the dice area of the union of the two areas and their intersection. Our



Convex hull Alpha shape

Figure 5.13: Convex hull to alpha-shape comparison.

preliminary results show that our proposed metric works well for small document length datasets like TWITTER, but not for larger ones. Figure 5.12 shows a schematic illustration of our new proposed idea.

The convex hull distance has several intriguing properties:

1. it is hyper-parameter free and straight-forward to understand and use;
2. it is highly interpretable as the areas covered by the two documents can be seen as the topics they hide.
3. it is very effective, as complexity remains low in small dimensions.

On the other hand, there are some limitations that need to be studied and addressed. Computing the convex hull is only efficient in 2 or 3 dimensions, with a complexity of $\mathcal{O}(n \log n)$. Additionally, since the words can be far from each other, the convex hull may end up covering a huge area. Thus we could instead use the alpha shape (α-shape), which is a tighter shape covering all the words in a document and was presented by Edelsbrunner et al. (1983). You can see an illustrative example in Figure 5.13.

Our RepSet approach was based on the availability of label information, to learn the sets. We would also like to explore learning the weights in the bipartite matching setting in an unsupervised way. This can be achieved via the local reconstruction error, like Karasuyama and Mamitsuka (2017) and Wu et al. (2018).

Last, we plan to extend our RepSet approach are replace bipartite matching with optimal transport. As the optimal transport can be solved as a linear pro-

gramming problem, we can apply the same methodology as the RepSet one and learn the latent representations with the Word Mover's Distance, thus presenting a differentiable Supervised Word Mover's Distance, without involving any metric learning techniques.

6

CONCLUDING REMARKS

Text is ubiquitous and has introduced numerous challenging problems to the research community. This dissertation has focused on extracting more meaningful information from textual data and using them for the text classification task. We specifically explored how we can get better metrics out of graph-based representations of text, use better linguistic structures for regularization, along with designing new algorithms for structured sparsity, enhance existing techniques for distance computation between text documents and last developed new architectures for learning hidden representations based on distances.

6.1 SUMMARY OF CONTRIBUTIONS

We particularly:

- Create new graph-based representations to model text in more meaningful structures. This gave us the ability to create new metrics like TW-ICW, which were more effective for identifying important words. This is the first work to combine document, collection and label level graphs along with word embeddings as weights in a full framework for text classification.

- Extracted novel linguistic groups and use them for structured regularization. Moreover we created new models for group structured regularization, like the overlapping Group Orthogonal Matching Pursuit, which brought accuracy with large sparsity.

- Boost existing document comparison techniques and design a new representation learning model for set classification. First, we enhanced the Word Mover's Distance with bag-of-tricks to make the document comparison more effective and efficient. Next, representing the comparison of two documents with a weighted bipartite matching scheme, we created a novel network approach with end-to-end gradient-based learning for text classification.

In the next Section, we provide an overview of the main contributions of the thesis and discuss future research directions.

6.2 FUTURE WORK

In this section, we briefly present the additional research ideas that came to mind while conducting the research presented in this dissertation. Some of them are direct follow-ups of our published works while others are more general and might be worth another Ph.D. thesis. They were left unexplored as they would require a significant amount of effort and time.

TW-IDW Following the notion of TW-ICW, we could challenge the document independence assumption usually made in text mining and consider a collection of documents as a Graph-of-Documents instead of a Bag-of-Documents in order for instance to compute an alternative to IDF. We could then envisage exploring term weights based on TW-IDW rather than TF-IDF. This assumes in particular that we have a Graph-of-Documents, even for Web-scale datasets, which is a research issue in itself, especially if the document similarity capitalizes on the Graph-of-Words representation, e. g., using the graph kernel proposed in Nikolentzos et al. (2017b).

GRAPH NEURAL NETWORKS Having graph-based text representations as input, one can utilize graph neural networks, like work by Niepert et al. (2016). Another work by Gilmer et al. (2017), which involves neural message passing, can be easily applied for the task of text classification.

LINGUISTIC STRUCTURED ATTENTION In Chapter 4 we studied regularization in text classification. A current work that we investigate is a group lasso attention mechanism for deep learning architectures. Having for example nested groups taken from a syntactic parse tree, the attention mechanism can drop entire constituents. When groups are overlapping but nested, the proximal operators can be computed top-down efficiently (Jenatton et al., 2010).

ADVERSARIAL STRUCTURED REGULARIZATION Following adversarial methods for regularization (Miyato et al., 2017; Miyato et al., 2018), we would also like to extend the structured regularization methods for Generative Adversarial Net-

works (Goodfellow et al., 2014). Alternatively, we may use adversarial rules for NLP models, like recent work by Ribeiro et al. (2018), as a way of regularization.

GRADIENT-BASED LEARNING FOR WORD MOVER'S DISTANCE Last, an easy extension of our SetRepNN approach can be done by replacing bipartite matching with optimal transport. As the optimal transport can be solved as a linear programming problem, we can apply the same methodology as the SetRepNN one and learn the latent representations with the Word Mover's Distance, thus presenting a differentiable Supervised Word Mover's Distance, without involving any metric learning techniques.

6.3 EPILOGUE

Managing and understanding text has been a key element in the area of Natural Language Understanding. Throughout this dissertation we presented novel methods for representations, regularization and distances in text. Even though a lot of work has been done in the field, there are still unanswered questions and challenging problems that will further enlighten our knowledge about the great problem of modelling language.

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[2] Manu Aery and Sharma Chakravarthy. "InfoSift: Adapting Graph Mining Techniques for Text Classification." In: *FLAIRS*. 2005, pp. 277–282.

[3] Charu C Aggarwal and ChengXiang Zhai. "A survey of text classification algorithms." In: *Mining text data*. Springer, 2012, pp. 163–222.

[4] Rie Kubota Ando and Tong Zhang. "A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data." In: *Journal of Machine Learning Research* 6 (2005), pp. 1817–1853.

[5] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, Georgios Paliouras, and Constantine D. Spyropoulos. "An evaluation of naive bayesian anti-spam filtering." In: *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning*. 2000, pp. 9–17.

[6] Kubilay Atasu, Thomas Parnell, Celestine Dünner, Manolis Sifalakis, Haralampos Pozidis, Vasileios Vasileiadis, Michail Vlachos, Cesar Berrospi, and Abdel Labbi. "Linear-complexity relaxed word Mover's distance with GPU acceleration." In: *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE. 2017, pp. 889–896.

[7] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999. ISBN: 0-201-39829-X.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." In: *International Conference on Learning Representations (ICLR)* (2015).

[9] Sergey Bakin. "Adaptive regression and model selection in data mining problems." Ph.D. Canberra, Australia: The Australian National University, May 1999.

[10] Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. "Regularization techniques for fine-tuning in neural machine translation." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), pp. 1489–1494.

[11] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A neural probabilistic language model." In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[12] Brigitte Bigi. "Using Kullback-Leibler distance for text categorization." In: *European Conference on Information Retrieval*. Springer. 2003, pp. 305–319.

[13] Roi Blanco and Christina Lioma. "Graph-based Term Weighting for Information Retrieval." In: *Inf. Retr.* 15.1 (2012), pp. 54–92.

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[15] John Blitzer, Mark Dredze, and Fernando Pereira. "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification." In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. 2007, pp. 440–447.

[16] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large networks." In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (2008), P10008.

[17] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. "A convolutional neural network for modelling sentences." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. 2014.

[18] Florian Boudin. "A Comparison of Centrality Measures for Graph-Based Keyphrase Extraction." In: *International Joint Conference on Natural Language Processing (IJCNLP)*. 2013, pp. 834–838.

[19] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. ISBN: 0-521-83378-7.

[20] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers." In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.

[21] Georgios-Ioannis Brokos, Prodromos Malakasiotis, and Ion Androutsopoulos. "Using centroids of word embeddings and word mover's distance for biomedical document retrieval in question answering." In: *Proceedings of the 15th Workshop on Biomedical Natural Language Processing* (2016).

[22] Ana Margarida de Jesus Cardoso Cachopo. "Improving methods for single-label text categorization." In: *Instituto Superior Técnico, Portugal* (2007).

[23] Liqun Chen, Shuyang Dai, Chenyang Tao, Dinghan Shen, Zhe Gan, Haichao Zhang, Yizhe Zhang, and Lawrence Carin. "Adversarial Text Generation via Feature-Mover's Distance." In: *Advances in Neural Information Processing Systems* (2018).

[24] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. "Atomic decomposition by basis pursuit." In: *SIAM review* 43.1 (2001), pp. 129–159.

[25] Stanley F. Chen and Ronald Rosenfeld. "A survey of smoothing techniques for ME models." In: *IEEE Transactions on Speech and Audio Processing* 8.1 (2000), pp. 37–50.

[26] François Chollet et al. *Keras*. 2015.

[27] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. "Attention-based models for speech recognition." In: *Advances in neural information processing systems*. 2015, pp. 577–585.

[28] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[29] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. "Very deep convolutional networks for text classification." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Vol. 1. 2017, pp. 1107–1116.

[30] William Jay Conover and William Jay Conover. "Practical nonparametric statistics." In: (1980).

[31] Thomas Cover and Peter Hart. "Nearest neighbor pattern classification." In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[32] Robert H Creecy, Brij M Masand, Stephen J Smith, and David L Waltz. "Trading MIPS and memory for knowledge engineering." In: *Communications of the ACM* 35.8 (1992), pp. 48–64.

[33] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research." In: *InterJournal, Complex Systems* 1695.5 (2006), pp. 1–9.

[34] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport." In: *Advances in neural information processing systems*. 2013, pp. 2292–2300.

[35] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. "Information-theoretic metric learning." In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 209–216.

[36] Franca Debole and Fabrizio Sebastiani. "Supervised term weighting for automated text categorization." In: *Text mining and its applications*. Springer, 2004, pp. 81–97.

[37] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. "Indexing by latent semantic analysis." In: *Journal of the American Society for Information Science*. JASIS '90 41.6 (1990), pp. 391–407.

[38] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. "Frequent Substructure-Based Approaches for Classifying Chemical Compounds." In: *IEEE Transactions on Knowledge and Data Engineering* 17.8 (Aug. 2005), pp. 1036–1050. ISSN: 1041-4347.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *arXiv preprint arXiv:1711.07227* (2018).

[40] Inderjit S. Dhillon. "Co-clustering Documents and Words Using Bipartite Spectral Graph Partitioning." In: *KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2001, pp. 269–274.

[41] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[42] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. "On the shape of a set of points in the plane." In: *IEEE Transactions on information theory* 29.4 (1983), pp. 551–559.

[43] David Eppstein and Joseph Wang. "Fast Approximation of Centrality." In: *J. Graph Algorithms Appl.* 8 (2004), pp. 39–45.

[44] Günes Erkan and Dragomir R. Radev. "LexRank: Graph-based Lexical Centrality As Salience in Text Summarization." In: *J. Artif. Int. Res.* 22.1 (Dec. 2004), pp. 457–479. ISSN: 1076-9757.

[45] Santo Fortunato. "Community detection in graphs." In: *Physics reports* 486.3 (2010), pp. 75–174.

[46] Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. *A note on the group lasso and a sparse group lasso*. Tech. rep. Department of Statistics, Stanford University, 2010.

[47] Alexander Genkin, David D Lewis, and David Madigan. "Large-scale Bayesian logistic regression for text categorization." In: *Technometrics* 49.3 (2007), pp. 291–304.

[48] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural Message Passing for Quantum Chemistry." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 1263–1272.

[49] Amir Globerson and Sam T Roweis. "Metric learning by collapsing classes." In: *Advances in neural information processing systems*. 2006, pp. 451–458.

[50] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. "Neighbourhood components analysis." In: *Advances in neural information processing systems*. 2005, pp. 513–520.

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[52] Kristen Grauman and Trevor Darrell. "Approximate correspondences in high dimensions." In: *Advances in Neural Information Processing Systems*. 2007, pp. 505–512.

[53] Kristen Grauman and Trevor Darrell. "The pyramid match kernel: Efficient learning with sets of features." In: *Journal of Machine Learning Research* 8.Apr (2007), pp. 725–760.

[54] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[55] Zellig S Harris. "Distributional structure." In: *Word* 10.2-3 (1954), pp. 146–162.

[56] Samer Hassan, Rada Mihalcea, and Carmen Banea. "Random-Walk Term Weighting for Improved Text Classification." In: *ICSC*. 2007, pp. 242–249.

[57] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning*. Vol. 2. Springer, 2009.

[58] Magnus R. Hestenes. "Multiplier and gradient methods." In: *Journal of Optimization Theory and Applications* 4 (1969), 303—320.

[59] Arthur E. Hoerl and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." In: *Technometrics* 12.1 (1970), pp. 55–67.

[60] Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. "Supervised word mover's distance." In: *Advances in Neural Information Processing Systems*. 2016, pp. 4862–4870.

[61] David Hull. "Using statistical testing in the evaluation of retrieval experiments." In: *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1993, pp. 329–338.

[62] Dat Huynh, Dat Tran, Wanli Ma, and Dharmendra Sharma. "A new term ranking method based on relation extraction and graph model for text classification." In: *Proceedings of the Thirty-Fourth Australasian Computer Science Conference-Volume 113*. Australian Computer Society, Inc. 2011, pp. 145–152.

[63] Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. "Japanese text normalization with encoder-decoder model." In: *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*. 2016, pp. 129–137.

[64] Tommi Jaakkola and David Haussler. "Exploiting generative models in discriminative classifiers." In: *Advances in Neural Information Processing Systems*. 1999, pp. 487–493.

[65] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. "Group Lasso with Overlap and Graph Lasso." In: *Proceedings of the 26th International Conference on Machine Learning*. ICML '09. ACM, 2009, pp. 433–440. ISBN: 978-1-60558-516-1.

[66] Tony Jebara, Risi Kondor, and Andrew Howard. "Probability product kernels." In: *Journal of Machine Learning Research* 5.Jul (2004), pp. 819–844.

[67] Rodolphe Jenatton, Julien Mairal, Francis Bach, and Guillaume Obozinski. "Proximal methods for sparse hierarchical dictionary learning." In: *Proceedings of the 27th International Conference on Machine Learning*. ICML '10. ACM, 2010, pp. 487–494.

[68] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. "Structured variable selection with sparsity-inducing norms." In: *Journal of Machine Learning Research* 12 (2011), pp. 2777–2824.

[69] Chuntao Jiang, Frans Coenen, Robert Sanderson, and Michele Zito. "Text classification using graph mining-based feature extraction." In: *Knowledge-Based Systems* 23.4 (2010), pp. 302–308.

[70] Thorsten Joachims. "Text categorization with Support Vector Machines: Learning with many relevant features." In: *Proceedings of the 10th European Conference on Machine Learning*. ECML '98. 1998, pp. 137–142.

[71] Rie Johnson and Tong Zhang. "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks." In: *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*. 2015, pp. 103–112.

[72] Rie Johnson and Tong Zhang. "Supervised and Semi-supervised Text Categorization Using LSTM for Region Embeddings." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 526–534.

[73] Rie Johnson and Tong Zhang. "Deep pyramid convolutional neural networks for text categorization." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 562–570.

[74] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. "Bag of Tricks for Efficient Text Classification." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, 2017, pp. 427–431.

[75] Masayuki Karasuyama and Hiroshi Mamitsuka. "Adaptive edge weighting for graph-based learning algorithms." In: *Machine Learning* 106.2 (2017), pp. 307–335.

[76] Dor Kedem, Stephen Tyree, Fei Sha, Gert R Lanckriet, and Kilian Q Weinberger. "Non-linear metric learning." In: *Advances in Neural Information Processing Systems*. 2012, pp. 2573–2581.

[77] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung-Hyon Myaeng. "Some Effective Techniques for Naive Bayes Text Classification." In: *IEEE Trans. Knowl. Data Eng.* 18.11 (2006), pp. 1457–1466.

[78] Seyoung Kim and Eric P. Xing. "Tree-guided group lasso for multi-task regression with structured sparsity." In: *Proceedings of the 27th International Conference on Machine Learning*. ICML '10. ACM, 2010, pp. 543–550.

[79] Sun Kim, Nicolas Fiorini, W John Wilbur, and Zhiyong Lu. "Bridging the gap: Incorporating a semantic similarity measure for effectively mapping PubMed queries to documents." In: *Journal of biomedical informatics* 75 (2017), pp. 122–127.

[80] Yoon Kim. "Convolutional Neural Networks for Sentence Classification." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1746–1751.

[81] Yuwon Kim, Jinseog Kim, and Yongdai Kim. "Blockwise sparse regression." In: *Statistica Sinica* (2006), pp. 375–390.

[82] Youngjoong Ko. "A study of term weighting schemes using class information for text classification." In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2012, pp. 1029–1030.

[83] Risi Kondor and Tony Jebara. "A kernel between sets of vectors." In: *Proceedings of the 20th International Conference on Machine Learning*. 2003, pp. 361–368.

[84] Eyal Krupka and Naftali Tishby. "Incorporating Prior Knowledge on Features into Learning." In: *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*. Vol. 2. AISTATS '07. 2007, pp. 227–234.

[85] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. "From word embeddings to document distances." In: *International Conference on Machine Learning*. 2015, pp. 957–966.

[86] Shibamouli Lahiri, Sagnik Ray Choudhury, and Cornelia Caragea. "Keyword and Keyphrase Extraction Using Centrality Measures on Collocation Networks." In: *CoRR* (2014).

[87] Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. "A Comprehensive Comparative Study on Term Weighting Schemes for Text Categorization with Support Vector Machines." In: *Special interest tracks and posters of the 14th international conference on World Wide Web (WWW)*. 2005, pp. 1032–1033.

[88] Man Lan, Chew-Lim Tan, Jian Su, and Yue Lu. "Supervised and Traditional Term Weighting Methods for Automatic Text Categorization." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 31.4 (2009), pp. 721–735.

[89] Leah S Larkey and W Bruce Croft. "Combining classifiers in text categorization." In: *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1996, pp. 289–297.

[90] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series." In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[91] Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Molding cnns for text: non-linear, non-consecutive convolutions." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015).

[92] Edda Leopold and Jörg Kindermann. "Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?" In: *Mach. Learn.* 46.1-3 (2002).

[93] Jiaxin Li, Ben M Chen, and Gim Hee Lee. "So-net: Self-organizing network for point cloud analysis." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9397–9406.

[94] Marina Litvak and Mark Last. "Graph-based Keyword Extraction for Single-document Summarization." In: *MMIES*. 2008, pp. 17–24.

[95] Jun Liu and Jieping Ye. "Moreau-Yosida Regularization for Grouped Tree Structure Learning." In: *Advances in Neural Information Processing Systems 23*. NIPS '10. Curran Associates, Inc., 2010, pp. 1459–1467.

[96] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. "Topical Word Embeddings." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2015, pp. 2418–2424.

[97] Aurelie C. Lozano, Grzegorz Swirszcz, and Naoki Abe. "Group Orthogonal Matching Pursuit for Logistic Regression." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS*. 2011, pp. 452–460.

[98] Wei Lu, Hai Leong Chieu, and Jonathan Löfgren. "A general regularization framework for domain adaptation." In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 950–954.

[99] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation." In: *EMNLP* (2015).

[100] Siwei Lyu. "A kernel between unordered sets of data: The gaussian mixture approach." In: *European Conference on Machine Learning*. Springer. 2005, pp. 255–267.

[101] Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. "Group Sparse CNNs for Question Classification with Answer Sets." In: *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)* (2017).

[102] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[103] James MacQueen et al. "Some methods for classification and analysis of multivariate observations." In: *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*. 1967, pp. 281–297.

[104] Julien Mairal, Rodolphe Jenatton, Francis Bach, and Guillaume Obozinski. "Network flow algorithms for structured sparsity." In: *Advances in Neural Information Processing Systems 23*. NIPS '10. Curran Associates, Inc., 2010, pp. 1558–1566.

[105] Stéphane G Mallat and Zhifeng Zhang. "Matching pursuits with time-frequency dictionaries." In: *IEEE Transactions on signal processing* 41.12 (1993), pp. 3397–3415.

[106] Fragkiskos D. Malliaros and Konstantinos Skianis. "Graph-based term weighting for text categorization." In: *Proceedings of the 1st International Workshop on Data Science for Social Media and Risk, IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM. 2015, pp. 1473–1479.

[107] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[108] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[109] Justin Martineau and Tim Finin. "Delta TFIDF: An Improved Feature Space for Sentiment Analysis." In: *ICWSM '09: Proceedings of the Third International Conference on Weblogs and Social Media*. 2009.

[110] Andre Martins and Ramon Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification." In: *International Conference on Machine Learning*. 2016, pp. 1614–1623.

[111] Andrew McCallum and Kamal Nigam. "A comparison of event models for Naive Bayes text classification." In: *Proceedings of the AAAI workshop on learning for text categorization*. AAAI '98. AAAI Press, 1998, pp. 41–48.

[112] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[113] Lukas Meier, Sara Van De Geer, and Peter Buhlmann. "The group lasso for logistic regression." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.1 (2008), pp. 53–71.

[114] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Texts." In: *Proceedings of EMNLP 2004*. Ed. by Dekang Lin and Dekai Wu. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 404–411.

[115] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[116] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." In: *ICLR Workshop* (2013).

[117] Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. "An Analysis of Statistical and Syntactic Phrases." In: *Proceedings of the 5th International Conference on Computer-Assisted Information Retrieval*. Vol. 97. RIAO '97. 1997, pp. 200–214.

[118] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. "Adversarial training methods for semi-supervised text classification." In: *International Conference on Learning Representations (ICLR)* (2017).

[119] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. "Virtual adversarial training: a regularization method for supervised and semi-supervised learning." In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[120] Sofia Mosci, Silvia Villa, Alessandro Verri, and Lorenzo Rosasco. "A Primal-Dual Algorithm for Group Sparse Regularization with Overlapping Groups." In: *Advances in Neural Information Processing Systems 23*. 2010, pp. 2604–2612.

[121] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., 2010.

[122] Andrew Y Ng, Michael I Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm." In: *Advances in neural information processing systems*. 2002, pp. 849–856.

[123] Maximillian Nickel and Douwe Kiela. "Poincaré embeddings for learning hierarchical representations." In: *Advances in neural information processing systems*. 2017, pp. 6338–6347.

[124] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning convolutional neural networks for graphs." In: *International conference on machine learning*. 2016, pp. 2014–2023.

[125] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. "Text Classification from Labeled and Unlabeled Documents using EM." In: *Machine Learning* 39.2-3 (2000), pp. 103–134.

[126] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. "Matching Node Embeddings for Graph Similarity." In: *Association for the Advancement of Artificial Intelligence*. 2017, pp. 2429–2435.

[127] Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. "Shortest-Path Graph Kernels for Document Similarity." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 1890–1900.

[128] Giannis Nikolentzos, Polykarpos Meladianos, Antoine Jean-Pierre Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. "Kernel graph convolutional neural networks." In: *International Conference on Artificial Neural Networks*. Springer, Cham. 2018, pp. 22–32.

[129] Guillaume Obozinski, Laurent Jacob, and Jean-Philippe Vert. "Group lasso with overlaps: the latent group lasso approach." In: *arXiv preprint arXiv:1110.0413* (2011).

[130] Stamatis Outsios, Konstantinos Skianis, Polykarpos Meladianos, Christos Xypolopoulos, and Michalis Vazirgiannis. "Word Embeddings from Large-Scale Greek Web content." In: *Spoken Language Technology (SLT)*. 2018.

[131] Georgios Paltoglou and Mike Thelwall. "A Study of Information Retrieval Weighting Schemes for Sentiment Analysis." In: *ACL '10: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. 2010, pp. 1386–1395.

[132] Bo Pang and Lilian Lee. "A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts." In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. 2004, pp. 271–278.

[133] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. *PyTorch*. 2017.

[134] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition." In: *Signals, Systems and Computers*. 1993, pp. 40–44.

[135] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python." In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[136] Tiago P Peixoto. "The graph-tool python library." In: *figshare* (2014).

[137] Shmuel Peleg, Michael Werman, and Hillel Rom. "A unified approach to the change of resolution: Space and gray-level." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7 (1989), pp. 739–742.

[138] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. "Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN." In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2018, pp. 1063–1072.

[139] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[140] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." In: *Proc. of NAACL*. 2018.

[141] M. J. D. Powell. "A method for nonlinear constraints in minimization problems." In: *R. Fletcher editor, Optimization* (1969), 283—298.

[142] Antti Puurula. "Cumulative progress in language models for information retrieval." In: *Proceedings of the Australasian Language Technology Association Workshop 2013 (ALTA 2013)*. 2013, pp. 96–100.

[143] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[144] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108.

[145] Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. "Linguistically regularized lstms for sentiment classification." In: *55th Annual Meeting of the Association for Computational Linguistics (ACL)* (2017).

[146] Rajat Raina, Andrew Y. Ng, and Daphne Koller. "Constructing Informative Priors Using Transfer Learning." In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. ACM, 2006, pp. 713–720. ISBN: 1-59593-383-2.

[147] Joel W. Reed, Yu Jiao, Thomas E. Potok, Brian A. Klump, Mark T. Elmore, and Ali R. Hurson. "TF-ICF: A New Term Weighting Scheme for Clustering Dynamic Data Streams." In: *ICMLA*. 2006, pp. 258–263.

[148] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora." English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[149] S Hamid Rezatofighi, Anton Milan, Ehsan Abbasnejad, Anthony Dick, Ian Reid, et al. "DeepSetNet: Predicting sets with deep neural networks." In: *Proceedings of the 2017 IEEE International Conference on Computer Vision*. 2017, pp. 5257–5266.

[150] S Hamid Rezatofighi, Anton Milan, Qinfeng Shi, Anthony Dick, and Ian Reid. "Joint learning of set cardinality and state distribution." In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2018.

[151] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Semantically equivalent adversarial rules for debugging nlp models." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2018, pp. 856–865.

[152] Stephen Robertson. "Understanding inverse document frequency: On theoretical arguments for IDF." In: *Journal of Documentation* 60 (2004), p. 2004.

[153] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, and Mike Gatford. "Okapi at TREC-3." In: *TREC '96: Proceedings of Text Retrieval Conference*. 1996, pp. 109–126.

[154] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. "Reasoning about entailment with neural attention." In: *International Conference on Learning Representations* (2016).

[155] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. "Stabilizing Training of Generative Adversarial Networks through Regularization." In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 2018–2028.

[156] Volker Roth and Bernd Fischer. "The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms." In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 848–855.

[157] François Rousseau and Michalis Vazirgiannis. "Graph-of-word and TW-IDF: New Approach to Ad Hoc IR." In: *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*. CIKM '13. San Francisco, California, USA: ACM, 2013, pp. 59–68. ISBN: 978-1-4503-2263-8.

[158] François Rousseau and Michalis Vazirgiannis. "Main Core Retention on Graph-of-Words for Single-Document Keyword Extraction." In: *ECIR*. 2015, pp. 382–393.

[159] François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. "Text Categorization as a Graph Classification Problem." In: *ACL '15: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. 2015.

[160] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The earth mover's distance as a metric for image retrieval." In: *International journal of computer vision* 40.2 (2000), pp. 99–121.

[161] Alexander M Rush, Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), 379—389.

[162] Gerard Salton. "The SMART retrieval system—experiments in automatic document processing." In: (1971).

[163] Gerard Salton and Chris Buckley. "Term-weighting approaches in automatic text retrieval." In: *Information Processing and Management* 24.5 (1988), pp. 513–523.

[164] Gerard Salton, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[165] Niek J Sanders. "Sanders-twitter sentiment corpus." In: *Sanders Analytics LLC* (2011).

[166] Ted Sandler, John Blitzer, Partha P. Talukdar, and Lyle H. Ungar. "Regularized learning with networks of features." In: *Advances in Neural Information Processing Systems 22 (NIPS)*. 2009, pp. 1401–1408.

[167] Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. "Deep belief nets for natural language call-routing." In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 5680–5683.

[168] Mark W. Schmidt and Kevin Murphy. "Convex structure learning in log-linear models: Beyond pairwise potentials." In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. AISTATS '10. JMLR Workshop and Conference Proceedings, 2010, pp. 709–716.

[169] Mark W. Schmidt, Glenn Fung, and Rómer Rosales. "Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches." In: *Proceedings of the 18th European Conference on Machine Learning*. 2007, pp. 286–297.

[170] Alexandra Schofield, Måns Magnusson, and David Mimno. "Pulling out the stops: Rethinking stopword removal for topic models." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Vol. 2. 2017, pp. 432–436.

[171] Fabrizio Sebastiani. "Machine Learning in Automated Text Categorization." In: *ACM Computing Surveys* 34.1 (Mar. 2002), pp. 1–47. ISSN: 0360-0300.

[172] Niloofer Shanavas, Hui Wang, Zhiwei Lin, and Glenn Hawe. "Centrality-Based Approach for Supervised Term Weighting." In: *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1261–1268.

[173] Masumi Shirakawa, Takahiro Hara, and Shojiro Nishio. "N-gram IDF: A Global Term Weighting Scheme Based on Information Distance." In: 2015.

[174] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgianis. "GraKeL: A Graph Kernel Library in Python." In: *arXiv preprint arXiv:1806.02193* (2018).

[175] Amit Singhal, Chris Buckley, and Mandar Mitra. "Pivoted Document Length Normalization." In: *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1996, pp. 21–29.

[176] Konstantinos Skianis, François Rousseau, and Michalis Vazirgiannis. "Regularizing Text Categorization with Clusters of Words." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1827–1837.

[177] Konstantinos Skianis, Maria-Evgenia G Rossi, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. "SPREADVIZ: Analytics and Visualization of Spreading Processes in Social Networks." In: *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1324–1327.

[178] Konstantinos Skianis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. "Fusing Document, Collection and Label Graph-based Representations with Word Embeddings for Text Classification." In: *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12), NAACL*. 2018, pp. 49–58.

[179] Konstantinos Skianis, Nikolaos Tziortziotis, and Michalis Vazirgiannis. "Orthogonal Matching Pursuit for Text Classification." In: *Proceedings of the Fourth Workshop on User-generated Text (W-NUT), Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[180] Konstantinos Skianis, Fragkiskos D. Malliaros, Nikolaos Tziortziotis, and Michalis Vazirgiannis. "Boosting Tricks for Word Mover's Distance." Manuscript. 2019.

[181] Konstantinos Skianis, Vlad Niculae, Guillaume Wisniewski, and Michalis Vazirgiannis. "Group Lasso for Linguistic Structured Attention." Manuscript. 2019.

[182] Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. "Rep the Set: Neural Networks for Learning Set Representations." Manuscript. 2019.

[183] Karen Sparck-Jones and RG Bates. "Research on automatic indexing, 1974-1976." In: (1977).

[184] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[185] Benjamin Stone, Simon Dennis, and Peter J Kwantes. "Comparing methods for document similarity analysis." In: *TopiCS, DOI* 10 (2010).

[186] John A Swets. "Information retrieval systems." In: *Science* 141.3577 (1963), pp. 245–250.

[187] Grzegorz Swirszcz, Naoki Abe, and Aurelie C Lozano. "Grouped Orthogonal Matching Pursuit for Variable Selection and Prediction." In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta. Curran Associates, Inc., 2009, pp. 1150–1158.

[188] Arthur Szlam, Karol Gregor, and Yann LeCun. "Fast approximations to structured sparse coding and applications to object classification." In: *Computer Vision–ECCV 2012* (2012), pp. 200–213.

[189] Jin Tao, Marco Cuturi, and Akihiro Yamamoto. "A Distance Between Text Documents based on Topic Models and Ground Metric Learning." In: *The 26th Annual Conference of the Japanese Society for Artificial Intelligence* (2012).

[190] Matt Thomas, Bo Pang, and Lillian Lee. "Get Out The Vote: Determining Support Or Opposition From Congressional Floor-Debate Transcripts." In: *Proceedings of EMNLP*. 2006, pp. 327–335.

[191] Robert Tibshirani. "Regression shrinkage and selection via the lasso." In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[192] Andrei Nikolaevich Tikhonov and Vasilii Iakovlevich Arsenin. *Solutions of ill-posed problems*. Winston, 1977. ISBN: 978-0-470-99124-4.

[193] Antoine Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. "Gowvis: a web application for graph-of-words-based text visualization and summarization." In: *Proceedings of ACL System Demonstrations* (2016), pp. 151–156.

[194] Joel A Tropp. "Greed is good: Algorithmic results for sparse approximation." In: *IEEE Transactions on Information theory* 50.10 (2004), pp. 2231–2242.

[195] Joel A Tropp and Anna C Gilbert. "Signal recovery from random measurements via orthogonal matching pursuit." In: *IEEE Transactions on information theory* 53.12 (2007), pp. 4655–4666.

[196] Cornelis Joost Van Rijsbergen. "Information retrieval." In: (1979).

[197] Lieven Vandenberghe and Stephen Boyd. "Semidefinite programming." In: *SIAM review* 38.1 (1996), pp. 49–95.

[198] Vladimir Naumovich Vapnik. "Principles of Risk Minimization for Learning Theory." In: *Advances in Neural Information Processing Systems 4*. 1991, pp. 831–838.

[199] Cédric Villani. *Optimal transport: old and new*. Vol. 338. Springer Science & Business Media, 2008.

[200] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. "Order matters: Sequence to sequence for sets." In: *International Conference on Learning Representations*. 2015.

[201] Rui Wang, Wei Liu, and Chris McDonald. "Corpus-independent Generic Keyphrase Extraction Using Word Embedding Vectors." In: 2015.

[202] Wei Wang, DiepBich Do, and Xuemin Lin. "Term Graph Model for Text Classification." In: *Advanced Data Mining and Applications*. Vol. 3584. 2005, pp. 19–30. ISBN: 978-3-540-27894-8.

[203] Kilian Q Weinberger and Lawrence K Saul. "Distance metric learning for large margin nearest neighbor classification." In: *Journal of Machine Learning Research* 10.Feb (2009), pp. 207–244.

[204] Nils Witt, Christin Seifert, and Michael Granitzer. "Explaining topical distances using word embeddings." In: *Database and Expert Systems Applications (DEXA), 2016 27th International Workshop on*. IEEE. 2016, pp. 212–217.

[205] Xuan Wu, Lingxiao Zhao, and Leman Akoglu. "A Quest for Structure: Jointly Learning the Graph Structure and Semi-Supervised Classification." In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM. 2018, pp. 87–96.

[206] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. "Overlapping Community Detection in Networks: The State-of-the-art and Comparative Study." In: *ACM Comput. Surv.* 45.4 (Aug. 2013), 43:1–43:35. ISSN: 0360-0300.

[207] Pengtao Xie, Aarti Singh, and Eric P. Xing. "Uncorrelation and Evenness: a New Diversity-Promoting Regularizer." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 3811–3820.

[208] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. "Distance metric learning with application to clustering with side-information." In: *Advances in neural information processing systems*. 2003, pp. 521–528.

[209] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. "Show, attend and tell: Neural image caption generation with visual attention." In: *International conference on machine learning*. 2015, pp. 2048–2057.

[210] Liu Yang and Rong Jin. "Distance metric learning: A comprehensive survey." In: *Michigan State University* 2.2 (2006).

[211] Yiming Yang. "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval." In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc. 1994, pp. 13–22.

[212] Yiming Yang and Xin Liu. "A re-examination of text categorization methods." In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 42–49.

[213] Liang Yao, Chengsheng Mao, and Yuan Luo. "Graph Convolutional Networks for Text Classification." In: *Association for the Advancement of Artificial Intelligence* (2019).

[214] Ian En-Hsu Yen, Wei-Cheng Lee, Sung-En Chang, Arun Sai Suggala, Shou-De Lin, and Pradeep Ravikumar. "Latent feature lasso." In: *International Conference on Machine Learning*. 2017, pp. 3949–3957.

[215] Dani Yogatama and Noah A. Smith. "Linguistic Structured Sparsity in Text Categorization." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 786–796.

[216] Dani Yogatama and Noah A. Smith. "Making the Most of Bag of Words: Sentence Regularization with Alternating Direction Method of Multipliers." In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, 2014, pp. 656–664.

[217] Bo Yu, Zong-ben Xu, and Cheng-hua Li. "Latent Semantic Analysis for Text Categorization Using Neural Network." In: *Know.-Based Syst.* 21.8 (Dec. 2008), pp. 900–904. ISSN: 0950-7051.

[218] Lei Yuan, Jun Liu, and Jieping Ye. "Efficient methods for overlapping group lasso." In: *Advances in Neural Information Processing Systems 24*. NIPS '11. Neural Information Processing Systems, 2011, pp. 352–360.

[219] Ming Yuan and Yi Lin. "Model selection and estimation in regression with grouped variables." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1 (2006), pp. 49–67.

[220] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. "Deep sets." In: *Advances in Neural Information Processing Systems*. 2017, pp. 3391–3401.

[221] Omar Zaidan and Jason Eisner. "Modeling Annotators: A Generative Approach to Learning from Annotator Rationales." In: *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference*. 2008, pp. 31–40.

[222] Zexian Zeng, Yu Deng, Xiaoyu Li, Tristan Naumann, and Yuan Luo. "Natural Language Processing for EHR-Based Computational Phenotyping." In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2018).

[223] Meng Zhang, Yang Liu, Huan-Bo Luan, Maosong Sun, Tatsuya Izuha, and Jie Hao. "Building Earth Mover's Distance on Bilingual Word Embeddings for Machine Translation." In: *Association for the Advancement of Artificial Intelligence*. 2016, pp. 2870–2876.

[224] Tong Zhang. "On the Consistency of Feature Selection using Greedy Least Squares Regression." In: *Journal of Machine Learning Research* 10 (2009), pp. 555–568.

[225] Tong Zhang and Frank J Oles. "Text categorization based on regularized linear classification methods." In: *Information retrieval* 4.1 (2001), pp. 5–31.

[226] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." In: *Advances in neural information processing systems*. 2015, pp. 649–657.

[227] Ye Zhang, Matthew Lease, and Byron C Wallace. "Exploiting Domain Knowledge via Grouped Weight Sharing with Application to Text Categorization." In: *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)* (2017).

[228] Hui Zou and Trevor Hastie. "Regularization and Variable Selection via the Elastic Net." In: *Journal of the Royal Statistical Society: Series B* 67.2 (2005), pp. 301–320.

ACRONYMS

Acc	Accuracy ... 12, 35
AI	Artificial Intelligence ... 1
ApproxRepSet	Approximate Set Representation Neural Network ... 93, 100, 101
BM	Bipartite Matching ... 93, 95
BoW	Bag-of-Words ... 2, 18, 43
CL	Computational Linguistics ... 1
CNN	Convolutional Neural Network ... 18, 21, 78
EMD	Earth Mover's Distance ... 78, 79
F1	F1-score ... 13, 35
FN	False Negative ... 11
FP	False Positive ... 11, 12
GOMP	Group Orthogonal Matching Pursuit ... 63, 64
GoW	Graph-of-Words ... 2, 17, 20, 27, 43, 59, 60
ICW	Inverse Collection Weighting ... 19, 22, 30–32
IDF	Inverse Document Frequency ... 3, 20, 24, 30
IR	Information Retrieval ... 19, 21–23, 28
KE	Keyword Extraction ... 23
kNN	k-Nearest Neighbors ... 10, 92
LDA	Latent Dirichlet Allocation ... 54, 55, 58, 59
LLSF	Linear Least Square Fit ... 10
LMNN	Large Margin Nearest Neighbors ... 86, 91
LR	Logistic Regression ... 10, 35, 66
LSA	Latent Semantic Analysis ... 10
LSI	Latent Semantic Indexing ... 53–55, 58–60, 62, 100
LSTM	Long Short Term Memory ... 78
LW	Label Weighting ... 19
MCML	Maximally Collapsing Metric Learning ... 88, 91
ML	Machine Learning ... 1, 12
NB	Naive Bayes ... 10
NCA	Neighborhood Component Analysis ... 82

NLG Natural Language Generation 1
NLP Natural Language Processing 1, 2, 21, 22, 52, 77
NLU Natural Language Understanding 1
OMP Orthogonal Matching Pursuit 63
P Precision 12
R Recall 12
RepSet Set Representation Neural Network 77, 93, 100
RWMD Relaxed Word Mover's Distance 79, 85
SVM Support Vector Machine 10, 35, 92
TC Text Classification or Text Categorization . . 8, 17–19, 21, 23, 25, 26, 28, 29, 35
TF Term Frequency 19, 21, 23, 24, 28, 29, 35, 36
TF-IDF Term Frequency-Inverse Document Frequency . . 10, 19, 20, 23–25, 28, 29, 35
TM Text Mining 1, 7
TN True Negative 11
TP True Positive 11, 12
TW Term Weighting 29, 32, 33, 35
TW-ICW Term Weighting-Inverse Collection Weighting 31, 35
TW-ICW-LW Term Weighting-Inverse Collection Weighting-Label Weighting 32, 36
TW-IDF Term Weighting-Inverse Document Frequency 29, 30, 33, 35
VSM Vector Space Model 9, 18
WMD Word Mover's Distance 4, 77–79, 81, 92, 100, 102

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`. The graphics in this dissertation are generated using the `Matlab` numerical computing environment, the `R` language, the `Ipe` extensible drawing editor and `pgf/tikz`. The bibliography is typeset using `biblatex`.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Nouvelles Représentations, Régularisations et Distances pour la Classification de Texte

Mots clés : fouille de texte, représentations, régularisation, distance

Résumé : Le texte a été le moyen dominant de stockage de données dans des systèmes informatiques et d'envoi d'informations sur le Web. Extraire du texte des représentations significatives a été un élément clé pour la modélisation de langage.
Le but de cette thèse est d'étudier les problèmes liés au traitement du langage naturel, comme l'apprentissage de la représentation, la régularisation de la classification des textes et la mesure de la distance entre les documents.
Dans la première partie de la thèse, nous avons étudié de nouvelles représentations à base de graphes pour le texte. Nous avons introduit ICW, une nouvelle métrique basée sur des graphiques au niveau de la collection afin de pénaliser les nœuds centraux, un peu comme l'IDF. Les rendements de TW-ICW et TW-ICW-LW sont comparables à ceux des classificateurs d'apprentissage en profondeur les plus récents pour la tâche de classification du texte.
Dans la deuxième partie de la thèse, nous nous sommes concentrés sur la régularisation pour le problème de l'apprentissage supervisé et plus spécifiquement pour la tâche de la classification du texte. Nous avons d'abord examiné comment divers groupes linguistiques existants peuvent aider de simples modèles de régression logistique pour la catégorisation de texte. Nous avons ensuite conçu une nouvelle version superposée de l'algorithme Orthogonal Matching Pursuit, une technique de sélection de variables gloutonne bien connue.
Dans la dernière partie de la thèse, nous étudions la mesure des distances entre les documents. Nous avons d'abord examiné les méthodes rapides permettant d'accroître la distance du populaire Word Mover 's Distance. Enfin, nous avons travaillé sur de nouvelles méthodes de graphes supervisés pour le calcul de la distance.

Title : Novel Representations, Regularization and Distances for Text Classification

Keywords : text mining, representations, regularization, distances

Abstract: Text has been the dominant way of storing data in computer systems and sending information around the Web. Extracting meaningful representations out of text has been a key element for modeling language.
The goal of this thesis is to study problems in the area of natural language processing, like representation learning, regularization in text classification and measuring distance between documents.
In the first part of the thesis, we have studied novel graph-based representations for text. We have introduced ICW, which is a new metric based on collection-level graphs in order to penalize central nodes, much like IDF. TW-ICW and TW-ICW-LW yield comparable to state-of-the-art deep learning classifiers results for the task of text classification.
In the second part of the thesis, we focused on regularization for supervised learning problem and more specifically for the task of text classification. First we examined how diverse existing linguistic groups can help simple logistic regression models for text categorization. Next, we designed a new overlapping version of the Orthogonal Matching Pursuit algorithm, a well-known greedy variable selection technique.
In the last part of the thesis, we study measuring distances between documents. First we examined fast methods for boosting the popular Word Mover's Distance. Last we worked on novel supervised graph-based methods for distance computation.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France