Programa de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Roger Alan Stein

An Analysis of Hierarchical Text Classification Using Word Embeddings

São Leopoldo, 2018

Roger Alan Stein

AN ANALYSIS OF HIERARCHICAL TEXT CLASSIFICATION USING WORD EMBEDDINGS

Dissertação apresentada à Universidade do Vale do Rio dos Sinos -– UNISINOS, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Orientadora:
Profª. Drª. Patrícia A. Jaques Maillard

Co-orientador:
Prof. Dr. João Francisco Valiati

São Leopoldo
2018

S819a Stein, Roger Alan

An analysis of hierarchical text classification using word embeddings / Roger Alan Stein – 2018.

57 f.: il.; 30 cm.

"Orientadora: Prof[a]. Dr[a]. Patrícia A. Jaques Maillard
Co-orientador: Prof. Dr. João Francisco Valiati".

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2018.

1. Classificação hierárquica. 2. Classificação textual. 3. Redes neurais (computação). 4. FastText. I. Título.

CDU 004

Dados Internacionais de Catalogação na Publicação (CIP)
Silvana Teresinha Dornelles Studzinski — CRB 10/2524

Roger Alan Stein

An Analysis of Hierarchical Text Classification Using Word Embeddings

Dissertação apresentada à Universidade do Vale do Rio dos Sinos -– UNISINOS, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 28 de março de 2018.

BANCA EXAMINADORA

Profª. Drª. Patrícia A. Jaques Maillard – UNISINOS

Prof. Dr. Sandro José Rigo – UNISINOS

Prof. Dr. Leandro Krug Wives – UFRGS

Profª. Drª. Patrícia A. Jaques Maillard (Orientadora)

Prof. Dr. João Francisco Valiati (Co-orientador)

Visto e permitida a impressão
São Leopoldo,

Prof. Dr. Sandro José Rigo
Coordenador PPG em Computação Aplicada

Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento PROSUC 88887.150315/2017-00.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Finance Code PROSUC 88887.150315/2017-00.

ACKNOWLEDGEMENTS

This work would not have been possible without the financial support of the Brazilian National Council for the Improvement of Higher Education (CAPES). For this reason, I am grateful to the local Scholarship Committee for this federal grant and for allowing me to undertake this research. I would also like to thank the Qualifying Board members for the insightful criticism on my research proposal, and the Applied Computing Graduate Program coordination for the prompt assistance in obtaining the dataset I used in this research.

I faced an unexpected setback halfway my journey when my original advisor left the University staff. Nevertheless, I have been very lucky to be surrounded by thoughtful people who helped me throughout the entire masters program and during those tumultuous times in particular. Therefore, I would like to express my gratitude to Prof[a]. Patricia Jaques Maillard for kindly accepting to carry on my project, for offering a fresh perspective about it, and for providing me with always keen comments.

Moreover, I am so deeply grateful to Prof. João Francisco Valiati for being my advisor since the beginning and continuing to closely supervise my work despite being no longer part of the University staff. Prof. Valiati has contributed in the making of this work more than I could ever give him credit for here, not only with his perspicacious indications, constructive recommendations, and thorough reviews, but also with his continued support and motivation—I greatly appreciate your dedication and consideration.

I am also grateful to my colleagues at UNISINOS, in particular Ricardo Gerhardt and Allan Barcellos, for the shared experiences and mutual growth; as well as to the Porto Alegre Machine Learning Meetup speakers and organizers, who sparked many ideas with their talks and conversations. I would like to thank my colleagues and managers at SAP for the fruitful discussions and for providing me with the time and environment to perform my research every time I needed.

Finally, nobody has been more important to me in the pursuit of this project than Deise Schell, my beloved partner in life, who experienced all of the ups and downs of my research—thank you for your continued patience and encouragement—and my dear, close family, Irma and Barbara Stein, who supported me unconditionally and understood my repeated absences.

"*Science is knowledge which we understand so well that we can teach it to a computer; and if we don't fully understand something, it is an art to deal with it.*"

— DONALD KNUTH, 1974

ABSTRACT

Efficient distributed numerical word representation models (word embeddings) combined with modern machine learning algorithms have recently yielded considerable improvement on automatic document classification tasks. However, the effectiveness of such techniques has not been assessed for the hierarchical text classification (HTC) yet. This study investigates application of those models and algorithms on this specific problem by means of experimentation and analysis. Classification models were trained with prominent machine learning algorithm implementations—fastText, XGBoost, and Keras' CNN—and noticeable word embeddings generation methods—GloVe, word2vec, and fastText—with publicly available data and evaluated them with measures specifically appropriate for the hierarchical context. FastText achieved an $\text{LCA}F_1$ of 0.871 on a single-labeled version of the RCV1 dataset. The results analysis indicates that using word embeddings is a very promising approach for HTC.

Keywords: Hierarchical classification. Text classification. Word embeddings. Convolutional neural networks. FastText.

RESUMO

Uma Análise de Classificação Hierárquica de Texto Usando *Word Embeddings*

Modelos eficientes de representação numérica textual (*word embeddings*) combinados com algoritmos modernos de aprendizado de máquina têm recentemente produzido uma melhoria considerável em tarefas de classificação automática de documentos. Contudo, a efetividade de tais técnicas ainda não foi avaliada com relação à classificação hierárquica de texto. Este estudo investiga a aplicação daqueles modelos e algoritmos neste problema em específico através de experimentação e análise. Modelos de classificação foram treinados usando implementações proeminentes de algoritmos de aprendizado de máquina—fastText, XGBoost e CNN (Keras)—e notórios métodos de geração de *word embeddings*—GloVe, word2vec e fastText—com dados disponíveis publicamente e avaliados usando métricas especificamente adequadas ao contexto hierárquico. Nesses experimentos, fastText alcançou um $\text{LCA}F_1$ de 0,871 usando uma versão da base de dados RCV1 com apenas uma categoria por tupla. A análise dos resultados indica que a utilização de *word embeddings* é uma abordagem muito promissora para classificação hierárquica de texto.

Palavras-chave: Classificação hierárquica. Classificação textual. Redes neurais (computação). FastText.

LIST OF ACRONYMS

BoW	bag of words
CNN	convolutional neural network
HC	hierarchical classification
hF_1	hierarchical F_1 measure
hP	hierarchical precision
hR	hierarchical recall
HTC	hierarchical text classification
IMDb	Internet Movie Database
LCA	lowest common ancestor
$\text{LCA}F_1$	lowest common ancestor F_1 measure
LCAP	lowest common ancestor precision
LCAR	lowest common ancestor recall
LCL	local classifier per level
LCN	local classifier per node
LCPN	local classifier per parent node
LDA	latent Dirichlet allocation
LSHTC	large scale hierarchical text classification
LSI	latent semantic indexing
LSTM	long short-term memory
MLNP	mandatory leaf node prediction
NLP	natural language processing
NMLNP	non-mandatory leaf node prediction
P	precision
R	recall
RcsNN	recursive neural network
RCV1	Reuters Corpus Volume 1
RNN	recurrent neural network
SVM	support vector machine
TC	text classification
TF-IDF	term frequency - inverse document frequency

CONTENTS

1 INTRODUCTION

Electronic text processing systems are ubiquitous nowadays—from instant messaging applications in smartphones to virtual repositories with millions of documents—and have created some considerable challenges to address users new information needs. One of such endeavors is classifying automatically some of this textual data so that information system users can more easily retrieve, extract, and manipulate information to recognize patterns and generate knowledge. Organizing electronic documents into categories has become of increasing interest for many people and organizations (KOLLER; SAHAMI, 1997; MANNING; RAGHAVAN; SCHÜTZE, 2008; INGERSOLL; MORTON; FARRIS, 2013; PROVOST; FAWCETT, 2013). Text classification (TC)—a.k.a. text categorization—is the field that studies solutions for this problem, and uses a combination of knowledge areas such as Information Retrieval, Artificial Intelligence, Natural Language Processing (NLP), Data Mining, Machine Learning, and Statistics. This is usually regarded as a supervised machine learning problem, where a model can be trained from several examples and then used to classify a previously unseen piece of text (SEBASTIANI, 2002; HAN; KAMBER; PEI, 2011).

TC tasks usually have two or a just few classes, for example, automatic email categorization, spam detection, customer request routing, fake news detection, etc. Classification tasks with a high number of possible target classes are studied as a further extension of the TC problem because they present some particular issues, which demand specific addressing or solutions. Many important real-world classification problems consist of a very large number of often very similar or overlapping categories that are organized into a class hierarchy or taxonomy (MANNING; RAGHAVAN; SCHÜTZE, 2008; SILLA JR.; FREITAS, 2011). This is where hierarchical classification (HC) arises: it is a particular type of structured classification problem, where the output of the classification algorithm must correspond to one or more nodes of a taxonomic hierarchy (SILLA JR.; FREITAS, 2011).

When applied to textual data, HC then obviously becomes hierarchical text classification (HTC). To illustrate with a real world analogy, HTC is similar to the task of the librarian who needs to find the right shelf for a book from its content. Some examples of large hierarchical text repositories are web directories (e.g., Best of the Web[1], DMOZ[2], Wikipedia topic classifications[3]), library and patent classification schemes (e.g., Library of Congress Classification[4], United States Patent Classification[5]), or the classification schemes used in medical applications (e.g., Medical Subject Headings (MeSH)[6]).

[1] <https://botw.org/>
[2] DMOZ—product of the Open Directory Project—was a web directory that used human editors to organize websites. It was closed on March 14, 2017 (SULLIVAN, 2017).
[3] <https://en.wikipedia.org/wiki/Category:Main_topic_classifications>
[4] <https://www.loc.gov/aba/cataloging/classification/>
[5] <https://www.uspto.gov/web/patents/classification/selectnumwithtitle.htm>
[6] <https://meshb.nlm.nih.gov/treeView>

1.1 Motivation and Justification

Many organizations can benefit from automatically classifying documents. For example, law firms can easily place and locate relevant cases (THOMPSON, 2001; SCHWEIGHOFER; RAUBER; DITTENBACH, 2001; GOVERNATORI, 2009), IT service providers can identify customer needs from incident tickets (ZENG et al., 2014; ECKSTEIN; KUEHL; SATZGER, 2016), medical organizations can categorize reference articles (TRIESCHNIGG et al., 2009; TSATSARONIS et al., 2015; PENG et al., 2016). Some of these examples already take advantage of hierarchical classification structures. Therefore, improvements within the HTC area can have a wide and considerable impact on many applications and areas of knowledge.

Investigation towards efficient methods to build classification models is of fundamental importance in this context. If a model cannot take advantage of all the training data available or cannot be inducted within a reasonable time, it may not offer an acceptable effectiveness, which in turn may not suit the application needs. The HTC problem poses some particular challenges, and while many classification algorithms are likely to work well in problems with only two or a small number of well-separated categories, accurate classification over large sets of closely related classes is inherently difficult (MANNING; RAGHAVAN; SCHÜTZE, 2008). To address that, some research has been applied to strategies that exploit the hierarchical structure in problems with a large number of categories. While some results suggest this approach shows some gain over working without using the taxonomy (MANNING; RAGHAVAN; SCHÜTZE, 2008; DUMAIS; CHEN, 2000) and is overall better than the flat classification approach (SILLA JR.; FREITAS, 2011), some conflicting HTC competition results still keep the question open whether hierarchical strategies really outperform flat ones (TSATSARONIS et al., 2015; PARTALAS et al., 2015). This is therefore a topic that still requires further examination to reach a consensus, as only recently evaluation measures for HTC problems have been better comprehended (COSTA et al., 2007; KOSMOPOULOS et al., 2015).

Moreover, in the recent years, some breakthroughs have been achieved in the machine learning and NLP fields, which have been improving the effectiveness of many TC systems. Such progress include two main topics: (1) efficient text representation in vector space models such as word embeddings (MIKOLOV et al., 2013a; PENNINGTON; SOCHER; MANNING, 2014) and (2) efficient classification algorithms implementations, e.g., softmax-based linear classifiers (JOULIN; GRAVE; MIKOLOV, 2017), scalable tree boosting systems (CHEN; GUESTRIN, 2016), and neural network variations (LECUN; BENGIO; HINTON, 2015). However, despite the close relationship between TC and HTC, the impact of those recent advancements have not been fully explored with regards to HTC yet.

1.2 Goals

The present work investigated whether and how some techniques that have recently shown to improve the results of TC tasks can be extended to have a positive impact on the HTC problem

through empirical experimentation and analysis. More specifically, this research has attempted to at least partially address the following main questions:

- How do recently developed text representation methods—GloVe, word2vec, and fastText—and efficient classification algorithms implementations—fastText, XGBoost, and Keras' CNN—that have recently boosted the flat text classification results improve the effectiveness of HTC?
- What are the classification models effectiveness difference when comparing traditional classification measures—e.g., flat F_1—against measures created specifically for hierarchical classification—e.g., hF_1 and $\text{LCA}F_1$?

1.3 Structure

The following chapter provides descriptions of formal HTC definitions (section 2.1), text representation schemes (section 2.2), and classification algorithms (section 2.3) that will be used for experimentation. Chapter 3 reviews relevant advancements within the HTC research, and the impact of recent techniques to similar classification tasks. Chapter 4 provides a detailed description of the experimental investigation along with its results and an analysis. Finally, Chapter 5 summarizes all findings and conclusions.

2 LITERATURE REVIEW

This chapter revisits the essential notions that are necessary to understand how the hierarchical structure of target categories can be used for TC, the methods used to represent text so that typical machine learning algorithms can be applied to it, and some of these algorithms.

2.1 Hierarchical Text Classification

While binary classification is the more general form of TC (SEBASTIANI, 2002), the current industry needs to extend far beyond this fundamental task, which is already challenging in its own way depending on the domain. Some TC tasks can have multiple classes, which can appear in different scenarios. If the classification problem allows for classes that are not mutually exclusive, i.e. if a text piece can belong to one, more than one, or no class at all, it is called an *any-of*, *multi-value*, or ***multi-label*** classification; on the other hand, if the classes are mutually exclusive, i.e. each document belongs to exactly one class, it is then called an *one-of*, *single-label*, *multinomial*, *polytomous*, or ***multi-class*** classification (MANNING; RAGHAVAN; SCHÜTZE, 2008). Throughout the present work, the terms in bold will be preferred.

If a multi-class task has a large set of categories, a hierarchical structure is usually present, and taking advantage of it during the learning and prediction processes defines what hierarchical classification is about (SILLA JR.; FREITAS, 2011). Koller & Sahami (1997) were some of the first researchers to notice that the classification schemes that existed at the time ignored the hierarchical structure and were often inadequate in cases where there is a large number of classes and attributes to cope with. This coincides with the emergent popularization of Internet directories such as Yahoo![7], which used to categorize the contents of the World Wide Web. In their proposed approach, they decompose the classification task into a set of simpler problems, and solve each one of them by focusing on a different set of features at each node.

As hierarchies were becoming ever more popular for the organization of text documents, researchers from the Institute of Informatics and Telecommunications - NCSR Demokritos in Athens, Greece and from the Laboratoire d'Informatique de Grenoble, France organized the Large Scale HTC (LSHTC) Challenge. LSHTC became a series of competitions to assess the effectiveness of classification systems in large-scale classification in a large number of classes, which occurred in four occasions (2009, 2011, 2012, and 2014), and set some benchmarks for the task (PARTALAS et al., 2015).

2.1.1 Problem Criteria and Solution Strategies

Different HC tasks may have different characteristics that affect how the problem is addressed, such as (1) the hierarchy type, (2) the required objective, and (3) the way the system

[7] Yahoo! (www.yahoo.com) was created as a directory of websites organized in a hierarchy in 1994. (BAEZA-YATES; RIBEIRO-NETO, 2011)

uses the hierarchy. As to (1) their type, hierarchical structures are typically trees or direct acyclic graphs—the main difference is that a node can have more than one parent node in the latter. The (2) task objective determines whether the classifier must always choose a leaf node—mandatory leaf node prediction (MLNP)—or can choose any node in any level—non-mandatory leaf node prediction (NMLNP) (SILLA JR.; FREITAS, 2011).

The most diverse characteristic relates to (3) how a classification system takes advantage of the hierarchy. Many approaches have been proposed to exploit the hierarchical structure of the target categories during the classification processes, and Silla Jr. & Freitas (2011) summarized them into three main clusters, namely (3.a) flat, (3.b) global, and (3.c) local approaches. The (3.a) flat classification for it ignores the hierarchy by "flattening" it to the leaf nodes level and works any usual multi-class classification algorithm during training and testing phases. In the (3.b) global approach, a.k.a. big-bang approach, a single classifier is trained while taking the hierarchy into account and may use a top-down strategy at the testing phase.

The (3.c) local classification approach, sometimes incorrectly referred as "top-down" approach, uses the hierarchy structure to build classifiers using local information, i.e. only the data that belongs to a particular node is considered to learn one or many classification models per each node. Silla Jr. & Freitas (2011) subdivide the local classification approach further into three subgroups depending on the way local information is used at the training phase: (3.c.I) local classifier per node (LCN) trains a binary classifier for each child node; (3.c.II) local classifier per parent node (LCPN) trains a multi-class classifier for each parent node; and (3.c.III) local classifier per level (LCL) trains a multi-class classifier for the entire hierarchy level. During the test phase, all systems built using the local classification approach use a top-down strategy, i.e. they predict a class at an uppermost level and then use that information to predict further under the candidates nodes from the previous step only in recursive manner until a leaf node is reached or the blocking criteria for a NMLNP is met.

2.1.2 Evaluation Measures

As hierarchical classification is inherently a multi-class problem, many researchers use traditional multi-class evaluation measures such as P (precision, i.e. the percentage of tuples predicted in a given class that actually belong to it), R (recall, i.e. the percentage of tuples correctly classified for a given class), and F_1 measure (a combination of precision and recall in a single measure) (HAN; KAMBER; PEI, 2011). Equations 2.1, 2.2, and 2.3 define these measures mathematically, where TP (true positive) is the number of tuples correctly predicted as belonging to category c, FP (false positive) is the number tuples predicted as belonging to category c that actually belong to other classes, and FN (false negative) is the number of tuples that actually belong to class c but where incorrectly classified in other classes. As HTC deals with many classes C, a single overall effectiveness value can only be obtained by averaging the mentioned measures, which can be done in two ways, namely, micro-average (average

of pooled contingency table) and macro-average (simple average over classes) (MANNING; RAGHAVAN; SCHÜTZE, 2008), which are calculated as per equations from 2.4 to 2.9.

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}, \qquad F_1 = \frac{2PR}{P + R} \tag{2.1, 2.2, 2.3}$$

$$\hat{P}_\mu = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} TP_c + FP_c}, \hat{R}_\mu = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} TP_c + FN_c}, \qquad \hat{F_{1\mu}} = \frac{2\hat{P}_\mu \hat{R}_\mu}{\hat{P}_\mu + \hat{R}_\mu} \tag{2.4, 2.5, 2.6}$$

$$\hat{P_M} = \frac{\sum_{c=1}^{C} P_c}{C}, \qquad \hat{R_M} = \frac{\sum_{c=1}^{C} R_c}{C}, \qquad \hat{F_{1M}} = \frac{2\hat{P_M}\hat{R_M}}{\hat{P_M} + \hat{R_M}}, \tag{2.7, 2.8, 2.9}$$

Nevertheless, these measures are actually inappropriate for HTC because they ignore the parent-child and sibling relationships between categories in a hierarchy, which is intuitively wrong because (1) assigning a tuple to a node near to the correct category is not as bad as assigning it to a distant node, and (2) errors in the upper levels of the hierarchy are worse than those in deeper levels (SUN; LIM, 2001; KIRITCHENKO et al., 2006; KOSMOPOULOS et al., 2015). As an attempt to resolve this problem, Sun & Lim (2001) proposed two HC measures: a category-similarity based one, which evaluates the effectiveness taking into consideration the feature vectors cosine distance between the correct and the predicted category; and a distance-based one, which assigns effectiveness considering the number of the links between the correct and the predicted category within the hierarchy structure.

Arguing that these methods are not applicable to directed acyclic graph (DAG) hierarchies nor multi-label tasks, and do not take the node level into consideration to measure the misclassification impact, Kiritchenko et al. (2006) propose an approach that extends the traditional precision and recall. Instead of considering only the actual and predicted nodes, their measures augment the objects under consideration by considering that each tuple belongs to all ancestors of the class it has been assigned to, except for the root node. The authors call these measures hierarchical precision (hP) and hierarchical recall (hR), which are suitable to calculate a hierarchical F_1 measure (hF_1). Although they claim having evidences that the new measures are superior to traditional ones, no experimental results have been provided.

Kosmopoulos et al. (2015) indicate such hierarchical versions of precision, recall, and F_1 excessively penalize errors in nodes with many ancestors. To address that, they propose a variation, in which they use the lowest common ancestor (LCA) as defined in graph theory—rather than the entire node ancestry as suggested by Kiritchenko et al. (2006)—to calculate precision (LCAP), recall (LCAR), and F_1 ($LCAF_1$) as indicated in equations from 2.10 to 2.12

$$\text{LCA}P = \frac{|\hat{Y}_{aug} \cap Y_{aug}|}{\hat{Y}_{aug}}, \quad \text{LCA}R = \frac{|\hat{Y}_{aug} \cap Y_{aug}|}{Y_{aug}}, \text{LCA}F_1 = \frac{2\text{LCA}P\text{LCA}R}{\text{LCA}P + \text{LCA}R} \tag{2.10, 2.11, 2.12}$$

where $\hat{Y}_{aug}$ and Y_{aug} are the augmented sets of predicted and true classes. Their $LCAF_1$ measure was studied empirically on datasets used by LSHTC and BioASQ[8] HTC competitions to conclude that "flat" measures are indeed not adequate to evaluate HC systems.

[8]BioASQ (<http://www.bioasq.org/>) is a challenge in large-scale biomedical semantic indexing and question answering that uses data from PubMed abstracts.

2.2 Text Representation

Although one can, for example, build a rule-based classifier upon a combination of keywords, most classification systems fundamentally consist of mathematical models that use machine learning-based approaches. Such models are designed to work with examples in the form of *m* tuples containing a fixed number of *n* attributes, ultimately represented as an *m*-by-*n* matrix. To comply with this constraint, any raw text that will be submitted to a classification procedure has to be transformed into a compliant representation, known as *vector space model* (SEBASTIANI, 2002; MANNING; RAGHAVAN; SCHÜTZE, 2008).

In most approaches, TC takes advantage of the techniques developed by the Information Retrieval community to address the document indexing problem in order to build such representation models. Some of these techniques generate a so-called document-term matrix, where each row corresponds to an element of the corpus and each column corresponds to a token from the corpus dictionary (MANNING; RAGHAVAN; SCHÜTZE, 2008), while others represent each document as a vector of an arbitrary size that contains a distribution of representing values, usually called topic models (BLEI, 2012). A third technique group, more recently developed, computes numerical document representation from distributed word representations previously derived with unsupervised learning methods (MIKOLOV et al., 2013b; LE; MIKOLOV, 2014; KUSNER et al., 2015; BALIKAS; AMINI, 2016). The following subsections present some of the most relevant text representation methods, which will be either used as benchmark or studied further with this investigation.

2.2.1 Document-term Matrix Approaches

The simplest way to represent text in a numerical format consists of transforming it to a vector where each element corresponds to a unique word and contains a value that indicates its "weight." Such a representation is known in the literature as the *bag of words* (BoW) model (MANNING; RAGHAVAN; SCHÜTZE, 2008). Transforming a document from raw text to BoW usually begins with some data cleansing and homogenization. In general, techniques such as tokenization (breaking a document into small chunks of text), downcasing (converting the text to lowercase), stemming (shortening a word to its base or root form), and stop words[9] removal are applied (INGERSOLL; MORTON; FARRIS, 2013). The next step concerns calculating the weight, which can be done using a wide variety of methods, but usually refers to a composite value derived from the term frequency (TF) and the inverse document frequency (IDF). Both TF and IDF can be calculated in many ways; the most common TF-IDF method attributes the term *t* and document *d* the weight as described in 2.13, where $tf_{t,d}$ is the number of times that term t appears in document d, N is the number of documents in the corpus, and df_t is the number of documents that contain term t (FELDMAN; SANGER, 2007).

[9] Stop words are extremely common words, such as propositions and articles, that have little value for indexing and other text processing tasks, e.g., the, of, in, at, etc. (MANNING; RAGHAVAN; SCHÜTZE, 2008)

$$TF\text{-}IDF_{t,d} = tf_{t,d} \cdot \log \frac{N}{df_t} \tag{2.13}$$

After calculating the BoW values for all documents in a corpus, each one is then cast into a sparse vector with as many elements as terms in the corpus dictionary to comply with the *m*-by-*n* matrix prerequisite. Although suitable and efficient for many text mining tasks, BoW models have a very high dimensionality, which poses considerable challenges to most classification algorithms, and ignores the word order completely (CRAIN et al., 2012). While there are some ways to at least partially tackle the former problem (using feature selection or dimensionality reduction, for example), addressing the latter is impossible as the ordering information loss is irreparable; as a result, classification algorithms will likely predict sentences such "China attacks France" and "France attacks China" as belonging to the same class, which is obviously wrong if one is trying to classify country belligerence. Furthermore, vector space representations are unable to cope synonymy and polysemy (MANNING; RAGHAVAN; SCHÜTZE, 2008).

2.2.2 Distributed Text Representation

Rather than having a separate attribute for each term in a corpus, a distributed text representation is a vector space model with an arbitrary number (usually around tens or a few hundreds) of columns that correspond to semantic concepts (CRAIN et al., 2012). Some of the most popular schemes in this approach, which is sometimes called *topic modeling*, are the latent semantic indexing (LSI) and latent Dirichlet allocation (LDA). LSI consists of a low-rank approximation of the document-term matrix built from it using singular value decomposition (SVD), and can arguably capture some aspects of basic linguistic notions such as synonymy and polysemy (DEERWESTER et al., 1990). LDA is a generative probabilistic model in which each document is modeled as a finite mixture of latent topics with Dirichlet distribution (BLEI; NG; JORDAN, 2003).

More recent approaches try to compose distributed text representation at document level from word representations in vector space—a.k.a. word vectors or word embeddings—generated with novel methods. Such methods include the continuous bag of words (CBoW) model, the continuous skip-gram model (CSG)—a.k.a. word2vec models[10] (MIKOLOV et al., 2013a)—and the global vectors model (GloVe) (PENNINGTON; SOCHER; MANNING, 2014). Word2vec models are trained using a shallow feed forward neural network that aims to predict a word based on the context regardless of its position (CBoW) or predict the words that surround a given single word (CSG) (MIKOLOV et al., 2013a). GloVe is a log-bilinear regression model that combines global co-occurrence matrix factorization (somehow similar to LSI) and local context window methods (PENNINGTON; SOCHER; MANNING, 2014). Classification algorithms then use the resulting word vectors directly or a combination of them as input to train

[10] Many researchers roughly refer to both CBoW and CSG models as *word2vec* models, which is the name of the software implementation provided by Mikolov et al. (2013a), which is available at <https://code.google.com/p/word2vec/>

models and make predictions (HUANG; QIU; HUANG, 2014; KIM, 2014; LAI et al., 2015; BALIKAS; AMINI, 2016). Moreover, some recently proposed classification algorithms incorporate the principles used to compute those word vectors into the classification task itself (LE; MIKOLOV, 2014; TAI; SOCHER; MANNING, 2015; JOULIN; GRAVE; MIKOLOV, 2017).

2.3 Classification Models

In a broad sense, classification is the process of attributing a label from a predefined set to an object, e.g., classifying an album according to its music genre. In the data mining context though, classification consists of a data analysis in two steps that (1) induces a model from a set of training tuples using statistical and machine learning algorithms that is able to (2) predict which class a previously unseen tuple belongs to (HAN; KAMBER; PEI, 2011). As it is a fundamental topic in many areas, classification has been a subject of intensive research over the last decades, which resulted in the proposal of many methods to generate, improve, and evaluate classifiers (MANNING; RAGHAVAN; SCHÜTZE, 2008). The following subsections provide an overview about some of them, namely, linear classifier, gradient tree boosting, and convolutional neural networks (CNN), for future reference in section 4.

2.3.1 Linear classifiers

A linear classifier assigns class c membership by comparing a linear combination of the features $\vec{w}^T\vec{x}$ to a threshold b, so that c if $\vec{w}^T\vec{x} > b$ and to $\bar{c}$ if $\vec{w}^T\vec{x} \leq b$. This definition can be extended to multiple classes by using either a multi-label (*any-of*) or a multi-class (*one-of*) method. In both cases, one builds as many classifiers as classes using a *one-versus-all* strategy. At the test time, a new test tuple is applied to each classifier separately. While all assigned classes are considered for the final result for the multi-label method, in the multi-class only the label with the maximum score b is assigned (MANNING; RAGHAVAN; SCHÜTZE, 2008, p.277–283). Rocchio[11], Naïve Bayes[12], and SVM[13] are examples of linear classifiers.

FastText is a model that essentially belongs to this group, but uses a combination of techniques to consider distributed word representation and word order while taking advantage of computationally efficient algorithms (JOULIN; GRAVE; MIKOLOV, 2017). It calculates embeddings in a similar way as the CBoW model does (MIKOLOV et al., 2013a), but with the label as the middle word and a bag of *n*-grams rather than a bag of words, which captures some information about the word order. The algorithm operates in two modes, supervised and

[11] Rocchio classification model uses centroids to calculate decision boundaries and classify a tuple according to the region it belongs to (MANNING; RAGHAVAN; SCHÜTZE, 2008).

[12] Naïve Bayes is a statistical model based on Bayes' theorem that makes predictions based on the probability that a tuple belongs to a class given its feature values (HAN; KAMBER; PEI, 2011).

[13] Support Vector Machine is a classification model that tries to find the hypothesis that minimizes the classification error based on the structural risk minimization principle by looking for the decision boundary that maximizes the distance between itself and the tuples that belong to either class (JOACHIMS, 1998).

unsupervised. In supervised mode, the documents are converted to vectors by averaging the embeddings that correspond to their words and used as the input to train linear classifiers with a hierarchical softmax function (GOODMAN, 2001). On the other hand, in unsupervised mode, fastText simply generates word embeddings for general purposes, then not taking classes into account.

2.3.2 Gradient Tree Boosting

A decision tree is a knowledge representation object that can be visually expressed as upside-down, tree-like graph in which every internal node designates a possible decision; each branch, a corresponding decision outcome; and a leaf node, the final result (class) of the decision set. If the leaf nodes contain continuous scores rather than discrete classes, it is then called a regression tree. In data mining, decision trees can be used as classification and regression models, and induced from labeled training tuples through methods such as the Hunt's algorithm (QUINLAN, 1986), which constructs the tree by recursively partitioning the data into smaller, purer subsets given a certain splitting criteria until either all remaining tuples belong to the same class, there are no remaining splitting attributes, or there are no remaining tuples for a given branch (HAN; KAMBER; PEI, 2011).

There are many ways to improve the tree induction algorithm by, for example, using different splitting criteria (gain ratio, information gain, Gini index, χ^2, etc.), pruning too specific branches, or using tree ensembles. Boosting is an ensemble method in which a classifier M_{i+1} is learned by "paying more attention" to the training tuples that were previously misclassified by M_i (HAN; KAMBER; PEI, 2011). The final classification is done by combining the votes of all M classifiers weighted by each model corresponding accuracy (HAN; KAMBER; PEI, 2011). Gradient boosting is a method that creates an ensemble of weak regression trees by iteratively adding a new one that improves the learning objective further through optimization of an arbitrary differentiable loss function (FRIEDMAN, 2001). A recent implementation of this method called XGBoost[14] combines computationally efficient principles—parallelism, sparsity awareness, cached data access—with additional improvement techniques, and has been allowing data scientists to achieve state-of-the-art results on many machine learning challenges (CHEN; GUESTRIN, 2016).

2.3.3 Neural Networks

A neural network (NN) is a set of units in the form of a DAG, where each unit node processes a function, and each connection has a weight associated with it (FREEMAN; SKAPURA, 1991; HAN; KAMBER; PEI, 2011; GOODFELLOW; BENGIO; COURVILLE, 2016). The most popular NN architecture is the multilayer feed-forward, in which the units are organized in

[14]The system is available as an open source package at <https://github.com/dmlc/xgboost>

three parts: the input layer, which receives the external data; the hidden layer, which might consist of many levels, indicating the depth of the network; and the output layer, which emits the network's prediction. NN's are most commonly trained by backpropagation, a method that iteratively updates the network connection weights to minimize the prediction errors (HAN; KAMBER; PEI, 2011).

Even though the interest on NN was less intense during some decades, it has been a topic of continuous research since its first appearance in the 1940s, and it has been drawing considerable attention due to the recent emergence of deep learning (DL) techniques over the last years (SCHMIDHUBER, 2015). Although having many hidden layers is a common characteristic of DL architectures, their key aspect is actually the fact that they allow for the representations of data with multiple levels of abstraction. This allowed DL to produce extremely promising results for various tasks in natural language understanding, particularly topic classification (LECUN; BENGIO; HINTON, 2015). Besides the general feed-forward neural network (FNN), a few specialized architectures are already used heavily in industry, including CNN and recurrent neural networks (RNN), which can scale to, for example, high-resolution images and long temporal sequences (GOODFELLOW; BENGIO; COURVILLE, 2016).

CNN is a specialization of FNN that employs convolution—a specialized kind of linear operation—rather than a matrix multiplication with connection weights. Its architecture usually consists of layers with three stages, namely, convolution, detection, and pooling. Despite its name, the convolution stage does not necessarily execute the title operation as mathematically defined; it applies a function over the input using a kernel that works as a kind of image filter resulting in a set of linear activations. The detection stage runs a nonlinear activation function over those previous results, usually a rectified linear activation. Finally, the pooling stage replaces the detection output with a summary statistic of nearby outputs, which might be used to make a final prediction or to connect to the next convolution layer (GOODFELLOW; BENGIO; COURVILLE, 2016).

3 RELATED WORK

The problem at hand has been a widely researched topic over the last two decades, and many approaches have been attempted towards the improvement of the classification results. At the same time, recent investigation on problems that bear some similarity with the HTC, such as binary TC (sentiment analysis, spam detection, etc) have experienced some rapid development with the usage of the representation and classification methods presented in sections 2.2 and 2.3.

This section aims to present past and current research status regarding HTC and some techniques used in related areas that can have an impact on this field as well, considering the similarity that it holds with other TC problems. Subsection 3.1 provides an overview about recent HTC research, sections 3.2 and 3.3 describe the advancements that other TC problems have seen in recent years, and finally subsection 3.4 critically analyzes all those studies and their relation to HTC.

3.1 Hierarchical Text Classification Research

Koller & Sahami (1997) proposed probably the first model that took the hierarchical structure of the target categories into consideration to build a classification system. It consisted of a set of Bayesian classifiers, one at each hierarchy node, which would direct a new incoming test tuple that made it through the parent nodes to the proper child node. Before being processed through the classifier, the text was submitted to a Zipf's Law-based[15] filter and encoded as a Boolean vector. Experimental results using Reuters-22173[16] showed a significantly higher accuracy then previous models due to (1) mainly the selection of features and (2) marginally the hierarchical disposition of the individual classifiers, as long as they are complex ones—i.e. the benefit was inconclusive while using Naïve Bayes model, but substantial with a more elaborated algorithm from the Bayesian family called KBD (SAHAMI, 1996).

Soon after that, Dumais & Chen (2000) used SVM to build an HTC model with two levels. The classification is based on a threshold, and considers parent and child nodes either in a Boolean decision function (LCN approach) or multiplying them (LCPN approach). In other words, the model would classify a tuple as belonging to a node if the calculated probability for it was higher than a user-specified value. The authors used SVM because it was considered an effective, efficient algorithm for TC, and experimented on a web content dataset with 350K records, which was a considerable amount for the time. The results showed a small F_1 improvement over flat models, but still statistically significant. On the other hand, a very recent study suggests that hierarchical SVM results do not considerably differ from the corresponding flat techniques (GRAOVAC; KOVAČEVIĆ; PAVLOVIĆ-LAŽETIĆ, 2017). Cesa-Bianchi,

[15] Zipf's Law is an empirical rule formulated by Zipf (1935) that states that the collection frequency cf_i of the ith most common term is proportional to $1/i$ (MANNING; RAGHAVAN; SCHÜTZE, 2008, p.82).

[16] <https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection>

Gentile & Zaniboni (2006) tried to combine both Bayesian and SVM models, but the results were unclear about any considerable advantage brought by their approach.

Ruiz & Srinivasan (2002) considered using an NN to create a model for HTC. Their collection of feedforward neural networks was inspired on Hierarchical Mixture of Experts (JORDAN; JACOBS, 1994), and consisted of a tree-structure composition of expert (linear function) and gating (binary function) networks trained individually. The experiments were executed on an excerpt of 233,455 records with 119 categories from the OHSUMED collection (HERSH et al., 1994). The data was filtered by stop words, stemmed using Porter's algorithm (PORTER, 1980), underwent feature selection through correlation coefficient χ^2, mutual information, and odds ratio, and then was finally submitted to the model. When comparing the results against a flat model, the hierarchical model results (as measured by an F_1 variation) are better, which indicates that exploiting the hierarchical structure increases effectiveness significantly. Nevertheless, the proposed approach is only equivalent to a Rocchio approach, which was used as benchmark.

In the realm of boosting methods, Esuli, Fagni & Sebastiani (2008) propose TreeBoost.MH, which is a recursive, hierarchical variant of AdaBoost.MH, a then well-known, multi-label algorithm that iteratively generates a sequence of weak hypotheses to improve upon it (SCHAPIRE; SINGER, 1998). The researchers experimented the method on Reuters-21578 (90 classes, $\sim$11K records), the RCV1[17] (103 classes, $\sim$800K records), and the ICCCFT[18] (79 classes, $\sim$1K records), and considered the same F_1 function variation as Ruiz & Srinivasan (2002) did for an effectiveness measure. Their conclusion is that the hierarchical AdaBoost.MH variant substantially surpasses the flat counterpart, in particular for highly unbalanced classes. Nonetheless, they mention that their approach is still inferior to SVM models, but make reservations regarding the validity of such a comparison.

The editions of the LSHTC Challenge brought many diverse approaches into the HTC area. Partalas et al. (2015) report on the dataset construction, evaluation measures, and results obtained by participants of the LSHTC Challenge. The most important dataset (used in 3 of the four editions) consisted of 2.8M records extracted from DBpedia[19] instances distributed among 325K classes. Instead of the original text from the DBpedia instance, each record consisted of a sparse vector with (feature, value) pairs resulting from a BoW processing. The challenge organizers used many evaluation measures, including accuracy, precision, recall, F_1 measure, and some hierarchically-specialized ones introduced over the years by Kosmopoulos et al. (2015), but not reported in this competition overview. The report summarizes the results by saying that flat classification approaches were competitive with the hierarchical ones, and highlights only a few that seem noteworthy, such as some models built upon k-Nearest Neighbor (kNN)[20] and Rocchio improvements. The 4th edition winning submission, in particular, consisted of

[17] Reuters Corpus Volume 1 (LEWIS et al., 2004)

[18] 2007 International Challenge on Classifying Clinical Free Text Using Natural Language Processing

[19] <http://wiki.dbpedia.org/>

[20] Nearest Neighbors classifiers are labor intensive classification methods that are based on comparing a given test tuple with training tuples that are similar to it (HAN; KAMBER; PEI, 2011, p.423)

an ensemble of sparse generative models extending Multinomial Naïve Bayes that combined document, label, and hierarchy level multinomials with feature pre-processing using variants of TF-IDF and BM25 (PUURULA; READ; BIFET, 2014).

Balikas & Amini (2016) elaborated an empirical study that employed word embeddings as features for large-scale TC. The researchers considered three versions with 1K, 5K, and 10K classes of a dataset with 225K records originally produced for the BioASQ competition, in which each record corresponds to the abstract, title, year, and labels of a biomedical article (TSATSARONIS et al., 2015). Despite using datasets with that high number of classes, these are not considered in a hierarchical fashion, which means the task consists of a flat, multi-label classification. The word embeddings were generated using the skip-gram model of word2vec based on 10 million PubMed[21] abstracts plus 2.5M Wikipedia documents in four sizes: 50, 100, 200, and 400 elements. The resulting embeddings of all words in a document abstract were combined using different functions—min, max, average, and a concatenation of these three—to compose a document representation, and the resulting vector was then used as input into an SVM classifier. The best results (as measured by F_1) are achieved by the concatenation of the outputs of the three composition functions, and are consistently better than the runner-up, the average function. Besides the way the word embeddings are combined, the vector size has a proportional effect on the classification effectiveness as well. The results, however, do not reach the baseline model, which is TF-IDF based SVM model. Nevertheless, when combining TF-IDF to the concatenated document distributed representations, the results are better than the TF-IDF alone by a small, but statistically significant margin.

3.2 Text Classification with Distributed Text Representations

While no breakthrough has occurred with the HTC task over the last years, other TC problems, on the other hand, have benefited from the recent great improvements on text representation using distributed vector space models. Over the recent years, many researchers used methods based on word embeddings to improve the accuracy of classification tasks such as sentiment analysis and topic classification. This section provides some examples from these other TC tasks that are somehow similar to HTC and took advantage from those advancements.

With regards to sentiment analysis, for example, Maas et al. (2011) created a method inspired on probabilistic topic modeling to learn word vectors capturing semantic term-document information with the final intend to tackle the sentiment polarization problem. They collected a dataset[22] with 100,000 movie reviews from the Internet Movie Database (IMDb)—25,000 labeled reviews for the classification task, 25,000 for the classification test, and 50,000 of unlabeled ones as additional data to build the semantic component of their model. The semantic

[21] <https://www.ncbi.nlm.nih.gov/pubmed/>

[22] The so-called Large Movie Review Dataset v1.0 has been widely used as a benchmark dataset for binary sentiment TC and is publicly available at <http://ai.stanford.edu/\~amaas/data/sentiment/>

component consisted of 50-dimensional vectors learned using an objective function that maximizes both the semantic similarities and the sentiment label. Their model outperformed other approaches, in particular when concatenated with BoW, when compared results upon the *polarity dataset v2.0*[23] (PANG; LEE, 2004).

Le & Mikolov (2014) proposed an unsupervised learning method that calculates vectors with an arbitrary length containing distributed representations of texts with variable length—the so-called *paragraph vectors*, which was highly inspired by the techniques used to learn word vectors introduced by Mikolov et al. (2013b). Such paragraph vectors can be used as features for conventional machine learning techniques, so the authors took the data collected by Maas et al. (2011) to calculate paragraph vectors, used them as inputs to a neural network to predict the sentiment, and compared the results against other approaches that used the same dataset. They reported a final result of 7.42% error rate, which they claim meant a new state-of-the-art result, with a significant relative error rate decrease in comparison to the best previously reported method.

Still on the sentiment analysis topic, however on a slightly different scenario, Tang et al. (2014) proposed the learning of Sentiment Specific Word Embedding (SSWE) by integrating the sentiment information into the loss function of the model and its application in a supervised learning framework for Twitter sentiment classification task. This is similar to the idea proposed by Maas et al. (2011), but uses a neural network rather then a probabilistic model. The authors used a partial version of a benchmark dataset used on SemEval[24] 2013 (NAKOV et al., 2013) with 6,251 positive/negative unbalanced records, and found that the SVM classification model built upon their SSWE has an effectiveness (macro-F_1) comparable with models created from state-of-the-art, manually designed features. Furthermore, they compared their SSWE with three other word embeddings—C&W[25] (COLLOBERT et al., 2011), word2vec (MIKOLOV et al., 2013a), and WVSA (Word Vectors for Sentiment Analysis) (MAAS et al., 2011)—to conclude that the effectiveness of word embeddings that do not directly take advantage of the sentiment information in the text—C&W and word2vec—are considerably lower than the others. Their study is just the beginning of a clear strategy trend in this topic: 7 out of the 10 top-ranked solutions for the SemEval-2016 Sentiment Analysis in Twitter Task incorporated either general-purpose or task-specific word embeddings in their participating systems (NAKOV et al., 2016). As an exponent of this trend, Vosoughi, Vijayaraghavan & Roy (2016) created a method to compute distributed representations for short texts using a long short-term memory (LSTM)[26] NN at a character-level. As their data source was the microblog Twitter,

[23] A publicly available dataset with 2,000 balanced, processed reviews from the IMDb archive.

[24] SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems organized by the Association for Computational Linguistics (ACL) <https://aclweb.org/aclwiki/SemEval_Portal>.

[25] C&W is a short that Tang et al. (2014) used for the method reportedly introduced by Collobert et al. (2011), which was not formally named by the authors, but apparently first appeared in Collobert & Weston (2008)

[26] The long short-term memory (LSTM) was originally proposed by Hochreiter & Schmidhuber (1997), uses "a memory cell which can maintain its state over time and non-linear gating units which regulate the information flow into and out of the cell" (GREFF et al., 2016), and is usually associated with the deep learning algorithms

they adequately named the method *Tweet2Vec*. The model was trained with 3 million records, which consisted of texts with at most 140 characters. To evaluate the quality of the resulting vectors, the authors used them to perform a polarity classification on the dataset provided on SemEval-2015 Task 10 subtask B competition[27]. The experiment consisted on extracting the vector representation from the texts on that dataset using their method and then train a logistic regression classifier on top of them. Their approach has reportedly exceed all others from that competition, and also surpassed Le & Mikolov (2014) *paragraph vector*, which was considered the state of the art in that context.

On the other hand, there are also examples of the usage of word embeddings on more general, multi category TC tasks. Huang, Qiu & Huang (2014) propose a method to learn so-called *document embeddings* directly in TC task, that aims to represent a document as a combination of the word embeddings of its words, which is learned using a neural network architecture. The authors use resulting network in two ways during the classification phase: the network itself as a classification model or the weights from one of its last hidden layers as the input for an SVM classifier. They test their methods on two datasets with 9 and 100 categories, and 17,014 and 13,113 training records, respectively—interestingly enough, the dataset with more categories was extracted from the LSHTC Challenge 4th edition, but ignored its hierarchical characteristic and used documents with a single label only. Although the authors report that their proposed architecture achieves better effectiveness on both datasets, the difference is only evident in one of them, and not enough statistical information is provided to support that claim.

Ma et al. (2015) use a Gaussian process approach to model the distribution of word embeddings according to their respective themes. The authors assume that probability of a document given a certain theme is the product of the probabilities of a word vector given the same theme. The classification task then becomes a problem of selecting the most probable Gaussian distribution that a document belongs to. The authors evaluate the model effectiveness with a dataset containing 10,060 training and 2,280 test short texts that belong to 8 unbalanced classes, which was previously used by other researchers. Their results show that the proposed method has a 3.3% accuracy gain over two other approaches that used (1) classical TF-IDF and (2) topic models estimated using latent Dirichlet allocation (LDA) as representation methods connected to MaxEnt classifiers, and suggest the accuracy increase occurs because "It is clear that using word embeddings which were trained from universal dataset mitigated the problem of unseen words." Nevertheless, accuracy is not an adequate metric for an unbalanced, multi-class classification (FAWCETT, 2006).

On another approach, Kusner et al. (2015) take advantage of the word embeddings to create a distance metric between text documents. Their proposed metric aims to incorporate the semantic similarity between word pairs—the lowest "traveling cost" (Euclidean distance) from a word to another within the word2vec embedding space—into a document distance function. The minimum cumulative cost to move from a document to another—the so-called *Word Mover's*

family (LECUN; BENGIO; HINTON, 2015).

[27] <http://alt.qcri.org/semeval2015/task10/>

Distance (WMD)—is then used to perform *k*NN document classification on eight real world document classification data sets. The resulting *k*NN classification model using WMD yields unprecedented low classification error rates when compared to other well-established methods such as latent semantic indexing (LSI) (DEERWESTER et al., 1990) and LDA (BLEI; NG; JORDAN, 2003).

Joulin, Grave & Mikolov (2017) built a classification model called fastText, already presented in section 2.3. The researchers ran experiments with two different tasks for evaluation, namely sentiment analysis and tag prediction. For sentiment analysis comparison, they used the same eight datasets and evaluation protocol as Zhang, Zhao & LeCun (2015), and found that fastText (using 10 hidden units, trained 5 epochs, with bigram information) accuracy is competitive with complex models, but needed only a fraction of time to process—the faster competitor took 24 minutes to train (some took up to 7 hours), while the worst case for fastText took only 10 seconds. Moreover, the authors claim they can still increase the accuracy by using more *n*-grams, for example with trigrams. For tag prediction evaluation, they used a single dataset[28] that contains information about images and focused on predicting the image tags according to their title and caption. When compared to Tagspace model (WESTON; BENGIO; USUNIER, 2011), fastText has a significantly superior effectiveness (as measured by precision-at-1)—because it is not only more accurate, but also uses bigrams—and runs more than an order of magnitude faster to obtain the model. To summarize, the fastText shallow approach seems to obtain effectiveness on par with complex deep learning methods, while being much faster.

3.3 Neural Networks for Text Classification

Neural network models have been investigated for TC tasks since mid-1990s (SCHÜTZE; HULL; PEDERSEN, 1995; WIENER et al., 1995). Nevertheless, at the same time that Sebastiani (2002) reports that some NN-based models using logistic regression provide some good results, he observes that they have a relative effectiveness slightly worse than many other models known at the time, e.g., SVM. This scenario would not change much during the following decade, such is that an evidence of NN models unpopularity in this area is the fact that Manning, Raghavan & Schütze (2008) do not even mention them in their classic textbook. Its resurgence among the TC research community would begin only in the late 2000's (ZHANG; ZHOU, 2006; TRAPPEY et al., 2006; YU; XU; LI, 2008) and maintain a steep increase over the following years, as will be shown in the upcoming paragraphs.

In a follow-up of the work described in their 2008's paper, Collobert et al. (2011) bring some new, radical ideas to the natural language processing area while deliberately disregarding a large body of linguistic knowledge to propose a neural network and learning algorithm that, contrary to the usual approach, is not task-specific, but can be applied to various tasks. In the authors' point of view, part-of-speech tagging (POS), chunking, named entity recognition

[28] YFCC100M dataset (THOMEE et al., 2016)

(NER), and semantic role labeling (SRL) problems can be roughly seen as assigning labels to words, so they build an architecture capable of capturing feature vectors from words and higher-level features through CNN to do that from raw text. Their training model architecture produces local features around each word of a sentence using convolutional layers and combines these features into a global feature vector, which is then fed into a standard layer for classification. They compare the results using this architecture against the state-of-the-art systems for each one of the four traditional NLP tasks just mentioned, and find that the results are behind the benchmark. To improve them, the authors create language models using additional unlabeled data that obtain feature vectors carrying more syntactic and semantic information, and use them as input for the higher-level layers of the architecture. This approach not only brings the results a lot closer to those benchmark systems', but demonstrates how important the use of word embeddings learned in an unsupervised way is. Not satisfied, Collobert et al. (2011) still use some multi-task learning schemes to have even more, better features, and eventually use some common techniques from the NLP literature as a last attempt to surpass the state-of-the-art systems. Their final standalone version of the architecture is a "fast and efficient 'all purpose' NLP tagger" that exceeds the effectiveness of the benchmark systems in all tasks, except for semantic role labeling, but only with a narrow margin.

Socher et al. (2013) proposed a model called Recursive Neural Tensor Network (RNTN) to compute compositional vector representations for phrases of variable length that are then used as features for sentiment classification. The input data consists of phrases parsed into binary trees with each leaf node corresponding to a word (and its corresponding vector). The model is based upon a standard recursive neural network (RcsNN), which computes parent vectors in a bottom-up fashion with different composition functions using the same softmax classifier; the RNTN, however, differs itself from those by introducing a tensor-based composition function. The authors introduced the Stanford Sentiment Treebank, a fully labeled parse trees phrase dataset, to accurately analyze the model at phrase level. The new model was compared to with other recursive methods, namely standard RcsNN and Matrix-Vector RcsNN (MV-RcsNN), and some commonly used methods—Naïve Bayes and SVMs using bag of words and bag of bigrams representations. The authors find that the recursive methods achieve better results than common methods, particularly because the bottom-up approach is able to capture negations at many levels in both positive and negative phrases. Finally, the proposed new method supplanted all others known at the time, and pushed the state-of-the-art accuracy on that dataset to 85.4%.

Still on sentiment classification studies, Kim (2014) reports on experiments with CNN trained upon distributed text representations. This approach is similar to previously mentioned architecture (COLLOBERT et al., 2011), but uses pre-trained word embeddings learned with word2vec and proposes a modification to allow for the use of both pre-trained and task-specific vectors by having multiple channels. The experiments included 7 datasets that not only related to sentiment polarity, but also considered subjectivity and question type classification, with number of classes between from 2 and 6, and dataset size ranging from 3.8 to 11.9 thousand records. The experiment results show that the authors' simple CNN with one convolution layer

only performs remarkably well despite little tuning of hyperparameters, surpassing the state-of-the-art methods in 4 out of the 7 analyzed tasks/datasets. It is not clear, however, what was the exact standard used to evaluate the effectiveness, nor whether the difference had a statistical significance, which is important as the tests related to 3 of the 4 superior scenarios were executed using a 10-fold cross-validation.

Johnson & Zhang (2015) also use CNN architecture, but instead of word embeddings, their model works on high-dimensional one-hot encoding vectors, i.e. each document is seen as sequence of dictionary-sized, ordered vectors with a single true bit each that corresponds to a given word. Their intention with this approach is capturing the word order within the network convolution. For evaluation purposes, the authors executed experiments on sentiment classification—IMDb dataset (MAAS et al., 2011)—and topic categorization—RCV1 dataset (LEWIS et al., 2004), disregarding the hierarchical structure –, and compared the results against SVM-based classifiers. Their CNN allegedly outperforms the baseline methods as measured by error rate, but this claim lacks some substantial statistical analysis. On a similar idea, but with a more minimalistic approach, the model proposed by Zhang, Zhao & LeCun (2015) accepts a sequence of encoded characters as input to a convolutional network to classify documents. In other words, the researchers deliberately disregard the grouping of letters in words, and transform text at character level into a 1,024 fixed length flow of vectors created with one-hot encoding. Since such kind of model requires very large datasets, in order to perform meaningful experiments, the authors had to build 8 of them, which had from 2 to 14 classes, number of records ranging from 120,000 to 3.6 million, and related to two main tasks, namely sentiment analysis and topic classification. To compare the models effectiveness, besides training their own new model, the authors also did so with models using (1) a multinomial logistic regression method built upon traditional text representation techniques and (2) a recurrent neural network using pre-trained word2vec word embeddings as input. In conclusion, they found that character-level convolutional networks are a feasible approach for TC, and confirmed that such models work best having large datasets for training.

Lai et al. (2015) combine recurrent and convolutional NN's to tackle the TC problem. At the model first level, a bi-directional recurrent structure captures the contextual information; at its second level, a max-pooling layer finds the best features to execute the classification. The model input consists of word embeddings that were pre-trained using the word2vec skip-gram method on Wikipedia dumps. For experimentation, the authors used 4 multi-class datasets with different sizes, and compared the model result in each dataset against the state-of-the-art approach for each dataset. Their model performs consistently well in all tested datasets, and even beats the best performing ones in half of the cases by a considerable difference, leading to the confirmation that NN-based approaches can compute an effective semantic text representation, and their conclusion that such approaches can also capture more contextual information of features than traditional methods.

3.4 Discussion and Considerations

Considering the works mentioned in this section, a few trends seem evident. The first thing to notice with regards to HTC research is that no reference dataset has apparently emerged over these two decades, and despite a few appear more often, there is no widely used standard. This obviously impedes effectively comparing the results of different studies in any way, as using the same data for experimentation is a prerequisite for any analysis with this intention. Although the Reuters' collections seemed to become popular at some point, they were disregarded by the LSHTC Challenge, which probably demanded a larger text collection. Nowadays even LSHTC dataset seems small, and its preprocessed format has actually become an inconvenient for researchers who intent to use distributed text representation. On a second note, no single effectiveness measure has been widely accepted yet as well. This is in part because of the variations within the HTC task itself (single- or multi-label), but also in part because it seems it took a long time for the community to evolve to a point when a thorough study about hierarchical classification evaluation could have been done. This fact not only poses a problem to compare the results among different studies, but also suggests that the comparison against flat models is not possible, as the measure for one problem is simply not the same as for the other. In other words, comparing the effectiveness of hierarchical classification against flat classification is not only inadequate, but also inaccurate, as the problem is different in its own nature (KOSMOPOULOS et al., 2015). In summary, the lack of consensus regarding a reference dataset and evaluation measures has negatively affected the HTC research development.

Many different methods have been applied to HTC, and the most representative ones have been referred, namely Bayesian models (KOLLER; SAHAMI, 1997; PUURULA; READ; BIFET, 2014), SVM (DUMAIS; CHEN, 2000; GRAOVAC; KOVAČEVIĆ; PAVLOVIĆ-LAŽETIĆ, 2017), NN (RUIZ; SRINIVASAN, 2002), boosting methods (ESULI; FAGNI; SEBASTIANI, 2008), Rocchio, and *k*NN (PARTALAS et al., 2015). This list is nonetheless far from exhaustive, as any text classification method (and any classifier in general, by extension), could be virtually used in this context. Nevertheless, neither a consistent effectiveness increase nor a breakthrough seem to have occurred over these two decades in the area. It is interesting to notice how Esuli, Fagni & Sebastiani (2008) consider the improvement achieved by their hierarchical model somehow surprising, as they would expect some effectiveness counter effect due to the unrecoverable incorrect classifications that occur in lower hierarchy levels, which is a known side-effect since Koller & Sahami (1997).

The fact that recent results still do not show a considerable difference between flat and hierarchical classification (PARTALAS et al., 2015; GRAOVAC; KOVAČEVIĆ; PAVLOVIĆ-LAŽETIĆ, 2017) sounds disquieting, to say the least, as one would expect that a specialized system should behave better than a generic one. Of course, the direct comparison does not hold, and should not be made. However, the proximity between flat and hierarchical classification makes it inevitable. Such comparison, on the other hand, makes the fact that HTC

researchers seem to have been paying little or no attention to the recent classification effectiveness improvements achieved using advances in other TC tasks also surprising. For example, considering the reports on TC competitions, while Partalas et al. (2015) do not even refer to the usage of word embeddings in LSHTC Challenges, Nakov et al. (2016) mention that most of the best systems performing sentiment analysis on Twitter used them in some way. However, after some consideration, it becomes clear that such advanced techniques could not have been applied to the LSHTC Challenge due to the way the dataset has been provided. Apparently, some data cleansing processing was so widely accepted around the competition years that the documents were heavily preprocessed (stemming/lemmatization, stop-word removal), and the idea of BoW was so well established that the organizers decided to deliver the dataset in a sparse vector format where each line corresponded to a document, and contained only token identifiers with its corresponding frequency. Although very popular and quite effective, this format misses some important lexical richness and lacks the word order, which is an overriding factor to detect semantic nuances.

It seems clear that word embeddings and other vector space models improve some TC schemes considerably. In the sentiment analysis task, techniques that either create vector space models in a supervised, polarity-induced manner (MAAS et al., 2011; SOCHER et al., 2013; TANG et al., 2014) or use general-purpose models (KIM, 2014; LE; MIKOLOV, 2014; LI et al., 2016) benefit from them. Similar advantage is reported in more general problems (HUANG; QIU; HUANG, 2014; MA et al., 2015), although some healthy skepticism is advisable regarding those reports as the evaluation methods are questionable. Nevertheless, the ideas behind word embeddings are undoubtedly advantageous for TC in many different ways, from calculating a distance metric for *k*-NN classifier (KUSNER et al., 2015) to transforming a word embedding learner into a classifier itself (JOULIN; GRAVE; MIKOLOV, 2017). Balikas & Amini (2016) mention they are aware that word embeddings are sometimes used as input for convolutional and recurrent neural network, but as their task concerns a large number of classes, they refrained from using them to avoid computational obstacles such as memory and processing overhead. The workaround they used, i.e. combining the word embeddings with simple arithmetic functions, yields good results, but still ignores the word order. All in all, despite its promising results, the effect of using word embeddings in HTC remains a great unknown, as no empirical evidence has been reported on it.

Analogously, it is evident that many TC and NLP tasks have been taking advantage of recent neural network architectures and deep learning improvements. Although Collobert & Weston (2008), Collobert et al. (2011) do not work with TC at sentence or document level, the ideas proposed therein seem significantly influential considering that many of the neural network architectures used today for that task had some inspiration taken from them[29]. Although Collobert et al. (2011) show that the use of CNN provides competitive results in more than one NLP classification task, the concepts that have been preached by them and others influenced many

[29]Those two papers combined had more than 3,500 citations as counted by Google Scholar by Jan 2017.

following researchers who later on started to reconsider NN models and find promising results (TANG et al., 2014; KIM, 2014). Their most important contribution to the present investigation is the indication that adequate word embeddings combined with appropriate classification NN's provide promising results.

On top of that, comparisons using simple feed-forward NN's against other methods have shown that the former are not only competitive, but even outperform the latter in many cases. This has been confirmed time and again with more complex architectures such as recursive NN's (SOCHER et al., 2013), recurrent NN's (ZHANG; ZHAO; LECUN, 2015), convolutional NN's (KIM, 2014), or a combination of them (LAI et al., 2015). All these works corroborate to the belief that a neural network is the most appropriate architecture to implement a state-of-the-art classification system. Nevertheless, this assumption lacks of empirical evidence when it comes to the hierarchical text context, as no report has been found specifically about it and it is doubtful that simple TC problems are adequate to evaluate deep neural networks representations, which in theory have power expected to provide much better final classification results (JOULIN; GRAVE; MIKOLOV, 2017).

4 EXPERIMENTS AND ANALYSIS

Experiments have designed and implemented to analyze the effectiveness of combining word embeddings as the text representation layer with modern classification algorithms applied to the HTC problem. After choosing an appropriate dataset for experimentation, which is described in section 4.1, a data flow was built to transform it depending on specific needs of each approach, as described in section 4.2, and classification models were trained using those techniques. Each model was used to predict the labels of tuples left aside during the training phase to evaluate its effectiveness, which was reported and analyzed in section 4.3.

4.1 Dataset

Since no dataset is widely used in the HTC research, choosing an appropriate dataset to perform HTC experiments becomes a somewhat hard task. The results from LSHTC Challenge would probably had been the best benchmark for comparison. However, as the LSHTC datasets are not available in a raw text format, they are inadequate for the purpose of this research. Therefore, corpora provided by Reuters (Reuters-22173 and RCV1) and PubMed (from BioASQ) were mainly considered. Although the PubMed collections have the advantage of containing a huge number of documents, they are ratter specialized for the medical area. This is considered as a downside for two reasons: (1) the results obtained within such a specific corpus might not generalize to other HTC tasks and (2) GloVe and word2vec pre-trained word vectors are general, which makes them inadequate for such a specific classification task. The Reuters collections have the advantages of including broader areas of knowledge—politics, economy, etc.—and the RCV1 (LEWIS et al., 2004) in particular has a reasonable size with regards to number of documents (around 800K) and categories (103). RCV1 is conveniently available as a collection of XML files and publicly accessible on request for research purposes[30]. Based on these pros and cons, RCV1 has been chosen as the experimental dataset, which contains one article per file with contents similar to example depicted in figure 1. The example shows that the document labels are identified within XML tag `<codes class="bip:topics:1.0">`.

The data preparation consisted in a few steps to adequate the dataset to the machine learning algorithms. First of all, those XML files were converted into text format to remove the hypertext tags. Since this analysis is particularly interested in the (single-label) multi-class problem, but most tuples had many labels (usually parent categories, as one can see in the example in figure 1), the number of tuples per category was calculated and only the least frequent category of each document was kept. This approach is based on the assumption that the least common label is the one that more specifically identifies the document. Some basic homogenization, such as lower case conversion and punctuation marks removal, was also performed.

The RCV1 hierarchy consists of 104 nodes (including root) distributed among 4 levels, with 22 of them having at least one child. The target classes are distributed among all levels except

[30] <http://trec.nist.gov/data/reuters/reuters.html>

```
<?xml version="1.0" encoding="iso-8859-1" ?>
...
<headline>Tylan stock jumps; weighs sale of
company.</headline>
...
<text><p>The stock of Tylan General Inc.  jumped
Tuesday after the maker of process-management
equipment said it is exploring the sale of
the company and added that it has already
received some inquiries from potential
buyers.</p>(...)</text>
...
<metadata>
...
<codes class="bip:topics:1.0">
  <code code="C15"> </code>
  <code code="C152"> </code>
  <code code="C18"> </code>
  <code code="C181"> </code>
  <code code="CCAT"> </code>
</codes>
...
</metadata>
```

Figure 1: An excerpt from a random XML file of the RCV1 dataset. (LEWIS et al., 2004)

```
Root
  CCAT - CORPORATE/INDUSTRIAL
    C11 - STRATEGY/PLANS
...
    C15 - PERFORMANCE
      C151 - ACCOUNTS/EARNINGS
        C1511 - ANNUAL RESULTS
      C152 - COMMENT/FORECASTS
...
  ECAT - ECONOMICS
    E11 - ECONOMIC PERFORMANCE
    E12 - MONETARY/ECONOMIC
      E121 - MONEY SUPPLY
    E13 - INFLATION/PRICES
      E131 - CONSUMER PRICES
      E132 - WHOLESALE PRICES
...
  GCAT - GOVERNMENT/SOCIAL
    G15 - EUROPEAN COMMUNITY
      G151 - EC INTERNAL MARKET
...
```

Figure 2: An excerpt from the RCV1 topics hierarchy. (LEWIS et al., 2004)

for the root, all nodes are potentially target classes, which indicates this matches an NMLNP task. In order to compare a flat classification against hierarchical LCPN approach, two data groups were created, one with all a simple train/test split using all RCV1 tuples, and a second one that was created by recursively stratifying subsets of the original dataset based on the parent nodes ("hierarchical split") and also further subdivided into train/test. Initial experiments with these datasets indicated the already expected incorrect classifications that occur with NMLNP in deeper hierarchy levels due to the models inability to stop the classification before reaching a leaf node or recovering from it (SILLA JR.; FREITAS, 2011). As a workaround, the subset stratification was re-executed by including a so-called virtual category (VC) using the tuples from the immediate parent node into the subset itself (except for the root node), as described by (YING et al., 2011). The final result of this hierarchical split is 22 datasets that contained between the entire dataset (root node) and less than 0.3% of that for training (node E14). The resulting datasets had a wide class imbalance variety, ranging from about 1:1 (node E51) to approximately 6000:1 (root node). Figure 2 shows an excerpt of the RCV1 topics hierarchy.

Besides using RCV1 and its hierarchy as the main elements for experimentation, general-purpose pre-trained word embeddings were also employed. The group responsible for word2vec published a dataset with around 3 million word vectors with 300 elements in length that were trained on about 100 billion words read from Google News dataset[31]. The authors of GloVe also published pre-trained versions of word vectors; for these experiments, a table with 2.2 million

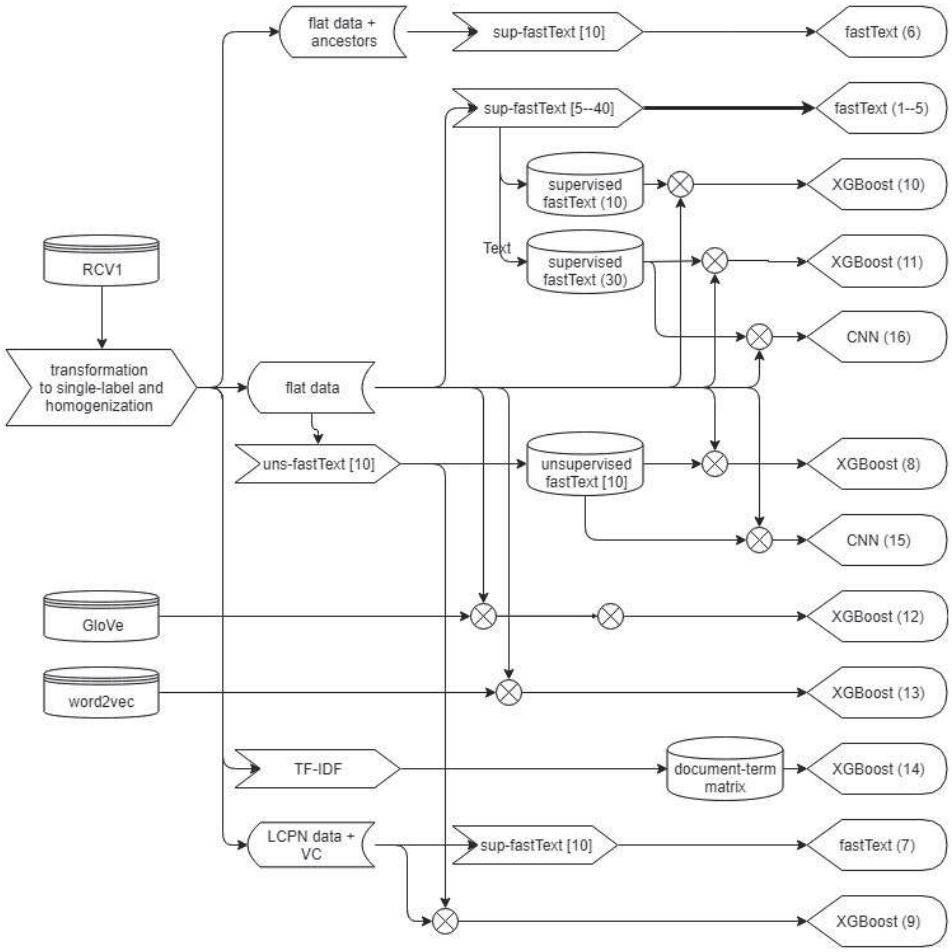
[31] https://code.google.com/archive/p/word2vec/

word vectors with 300 elements obtained from 840 billion words collected via Common Crawl[32] was used. Both pre-trained word vectors were used by at least one of the classification models analyzed here.

4.2 Classification Models and Data Flows for Experimentation

Out of the many possible classification models mentioned in Chapter 3, this analysis was concentrated on those three described in the list below. As each one of them has its own data input format, the corpus had to be processed in particular ways. Where feasible, models using both the LCPN and the flat approaches were generated—due to computational restrictions, the same could not be done for the CNN models. The following list describes the learning algorithms used with details about specific data preprocessing and variations attempted:

Figure 3: The flowchart depicts the processing routes used to build each classification model.



- FastText: This algorithm was used in two ways—as a supervised classification learner and as an unsupervised word embedding generator. As fastText is able to handle the raw

[32]https://nlp.stanford.edu/projects/glove/

text directly, no further pre-processing after the basic homogenization described in section 4.1 was necessary. In supervised mode, word embeddings with 5 different vector sizes—5, 10, 20, 30, and 40 elements—were explored to investigate how expanding the numerical distribution affects the final classification effectiveness. All word embeddings were generated considering bigrams. Based on the results from this exploratory phase, the 10-element word embeddings were used for the subsequent fastText experiments because no relevant effectiveness improvement was noticed with larger vectors. They were used to investigate 3 model construction strategy variations: (1) flat, (2) LCPN with VC, and (3) flat with ancestors. The *flat* approach, as described in subsection 2.1.1, ignores the hierarchy and works as a general multi-class classification algorithm, depicted by models from 1 to 5 in figure 3. The *LCPN with VC* creates a classification model for each of the 22 subsets that are used in a top-down strategy during the test phase and includes a virtual category that allows the classifier to keep the tuple at the present node—model 7. The *flat with ancestors* approach takes advantage of the fact that fastText is able to natively handle multi-class tasks—the data subsets were enhanced by adding the labels for all class ancestors to each tuple—model 6. This was an attempt to minimize another known negative side-effect of the LCPN approach—the inability to recover from an incorrect classification in an upper node. Furthermore, both supervised and unsupervised 10-element fastText word embeddings were recycled to experiment with the other classification models as well.

- XGBoost: In order to accommodate the distributed text representation in a format suitable to this algorithm, pre-trained word embeddings—GloVe, word2vec, and fastText separately—were combined to compose a document representation from the corresponding word embeddings average. In other words, a column-wise mean of the word embeddings that corresponded to the words from each document was taken to compound a distributed document representation. The resulting average vectors were used as input attributes to train the classifier. These data flows are represented by models from 8 to 13 in figure 3. Although this approach is similar to the one proposed by Balikas & Amini (2016) and also internally used by fastText, the present work used boosted trees algorithm implemented by Chen & Guestrin (2016) instead of an SVM or a linear classifier as the others did. The unsupervised fastText alternative was trained with both *flat* and *LCPN with VC* strategies (models 8 and 9) to assess how effective a flat approach is in comparison with a hierarchical one, and the supervised fastText alternative using vectors with 10 and 30 elements (models 10 and 11). Besides these architectures that use word embeddings, one more was implemented with TF-IDF representation (model 14), created from an all-lowercase, stemmed, punctuation- and stopword-free version of the RVC1 dataset. This has the purpose of comparing the traditional TF-IDF representation directly against the distributed text one. For all experiments, the XGBoost algorithm was set to use a softmax objective, which is the most adequate loss function among the ones XGBoost

has available for a multi-class task, and run it for 30 rounds, which proved to be enough to converge to minimum loss.

- CNN: Since this neural network specialization has a fixed input layer size, the corpus documents were padded to keep only a fraction of the input text, in a similar way as described by Kim (2014). An initial analysis on the corpus characteristics[33] indicated that keeping the last 600 words would have a minimal, tolerable effect on the final classification results. The Keras API (CHOLLET et al., 2015) was used to built a neural network with the following architecture: a frozen embedding layer that uses the fastText vectors either with 10 elements created in unsupervised mode (model 15) or with 30 elements created in supervised mode (model 16), then two convolution layers with rectified linear units (ReLU) and max pooling each (GOODFELLOW; BENGIO; COURVILLE, 2016), a densely-connected layer with ReLU activation and finally another densely-connected layer with softmax function. It was trained using categorical cross-entropy as the loss function over 10 epochs (LAI et al., 2015).

4.3 Results and Analysis

During the dataset preparation, a holdout method (HAN; KAMBER; PEI, 2011) was employed to reserve a test data subset, which was used to evaluate the models effectiveness according with the methods described in subsection 2.1.2. Besides the traditional flat classification measures—precision, recall, and F_1—their hierarchical and LCA versions were used to assess the models' effectiveness. (KOSMOPOULOS et al., 2015) have shown consistent results to support that $\text{LCA}F_1$ is the most appropriate measure for HTC evaluation. HEMkit[34] was used to calculate the hierarchical and LCA metrics, which are shown in table 1.

First and foremost, the results show an obvious, considerable numerical difference between flat and both hierarchical measures. It contributes to the ever growing understanding that flat measures are not adequate for the hierarchical context, as they insinuate a classification effectiveness well below the actual results. At the same time, there is a noticeable correlation between both hierarchical measurement methods with 0.988 Pearson coefficient between all corresponding pairs. This strong association may have occurred because the RCV1 hierarchy is only four levels deep while the main improvement offered by LCA measures in comparison to other hierarchical measures is that they prevent over-penalizing errors inflict nodes with many ancestors. This indicates that traditional hierarchical measures are enough in low hierarchical level classification scenarios—they might be even preferred in such cases as they are a simpler, computationally cheaper option.

[33] The pre-processed RCV1 training subset prepared for this work had an average document length of 261.57 words with 90 as the mode. Approximately 6% of the corpus has more than 600 words only.

[34] HEMkit is a software tool provided by the BioASQ team that performs the calculation of a collection of hierarchical evaluation measures.

#	Classifier	Hierarchy strategy	Text Representation Type	Size	Flat (macro averaged) P	R	F_1	Hierarchical P	R	F_1	LCA P	R	F_1
1	fastText	flat	sup-fastText	5	0.575	0.352	0.437	0.866	0.865	0.864	0.828	0.827	0.826
2	fastText	flat	sup-fastText	10	0.657	0.460	0.540	0.892	0.892	0.891	0.862	0.863	0.860
3	fastText	flat	sup-fastText	20	0.700	0.488	0.575	0.899	0.900	0.898	0.872	0.872	0.870
4	fastText	flat	sup-fastText	30	0.700	0.491	0.577	0.900	0.901	0.899	0.873	0.874	⋆0.871
5	fastText	flat	sup-fastText	40	0.698	0.490	0.576	0.900	0.900	0.899	0.873	0.873	⋆0.871
6	fastText	flat + ancestors	sup-fastText	10	0.588	0.103	0.175	‡0.963	0.602	0.731	‡0.892	0.453	0.587
7	fastText	LCPN + VC	sup-fastText	10	0.325	0.475	0.386	0.747	†0.946	0.821	0.631	†0.933	0.720
8	XGBoost	flat	uns-fastText	10	0.188	0.087	0.119	0.584	0.556	0.561	0.472	0.449	0.449
9	XGBoost	LCPN + VC	uns-fastText	10	0.041	0.054	0.046	0.328	†0.905	0.472	0.229	†0.804	0.345
10	XGBoost	flat	sup-fastText	10	0.445	0.384	0.412	0.837	0.837	0.835	0.795	0.796	0.792
11	XGBoost	flat	sup-fastText	30	0.485	0.401	0.439	0.842	0.842	0.840	0.801	0.801	0.798
12	XGBoost	flat	GloVe	300	0.182	0.720	0.290	0.830	0.822	0.824	0.787	0.768	0.771
13	XGBoost	flat	word2vec	300	0.189	0.852	0.310	0.835	0.826	0.828	0.792	0.774	0.777
14	XGBoost	flat	TF-IDF	∼340k	0.581	0.530	0.555	0.892	0.891	0.884	0.833	0.833	0.824
15	CNN	flat	uns-fastText	10	0.456	0.329	0.382	0.782	0.766	0.769	0.677	0.666	0.665
16	CNN	flat	sup-fastText	30	0.483	0.359	0.412	0.793	0.790	0.787	0.690	0.692	0.684

Table 1: Performance, Recall and F_1 measures in flat, hierarchical and LCA versions per classifier, hierarchical strategy, and word embedding. Legend: ⋆Highest $\text{LCA}F_1$. †Remarkable recall improvement. ‡Remarkable precision improvement.

With regards to classification effectiveness, fastText algorithm using no hierarchical information (flat approach) is surprisingly prominent among the models studied, and surpassed all hierarchy-aware models when measuring it by $\text{LCA}F_1$—highlighted with a star table 1—reaching a maximum value with 30 word embeddings or more. It is remarkable that even 10-element vectors with very limited representational power can achieve what is considered a somewhat high effectiveness with $\text{LCA}F_1$ greater than 0.85, which is superior to any other classifier analyzed in this research. This might come from the fact that, when used in supervised mode, fastText uses the class label as learning objective, which results in word embeddings that specifically reflect the concepts behind the classes distribution. This hypothesis is supported by the fact that fastText word embeddings created in supervised mode yielded effectiveness gains over all other unsupervised ones.

The results also indicate that using LCPN with VC consistently increases the hierarchical recall—marked with a dagger in table 1. This suggests that combining local classifiers with this circular strategy seems to increase the fraction of tuples placed in the right hierarchy branch over the total amount of tuples that should actually be there. However, this bouncing approach comes at a cost of decreasing the precision because some tuples get stuck at a certain level while they should be further classified at a deeper level node. On the other hand, using ancestors greatly increased the hierarchical precision—marked with a double dagger in the same table—which increases the chance that a prediction for a given hierarchy branch actually belongs to it. In summary, in both cases where flat models can be compared against their hierarchy-aware counterparts, the former are superior with regards to F_1, but the latter might still suit better a particular task depending on the specific user information needs.

The XGBoost results indicate that tree boosting methods associated with pre-trained word embeddings are a feasible combination to achieve reasonable classification results, with $\text{LCA}F_1$ close to 0.8. However still fall behind equivalent a model using TF-IDF representation, which achieved the second highest $\text{LCA}F_1$—0.824. Moreover, the word embedding systems seem to

depend on the word embeddings quality to a very great extent. Although word2vec embeddings have a slight advantage over GloVe's, the difference is not significant. Nevertheless, both word embeddings are clearly superior to those generated using the unsupervised fastText word embeddings. This most likely happens because those unsupervised fastText word embeddings were trained with the RCV1 train dataset only—this is a very small fraction of the amount of data that the other pre-trained word embeddings had been provided with. On the other hand, XGBoost achieved fair results when using supervised fastText word embeddings even using the same amount of data. This contributes to the understanding that even short word embeddings specifically generated during the classification task are the most adequate representation for this problem.

The CNN model provided somewhat interesting results despite that it had neither a thoroughly designed architecture nor fine-tunned hyper parameters. When using fastText 10-element word embeddings generated in unsupervised mode, it was considerably superior to both XGBoost models that used the same input data. Moreover, it achieved similar results using either unsupervised or supervised fastText word embeddings, which denotes that it is less dependent on the word embeddings quality. Therefore, this is considered this a quite promising approach. Its learning phase, however, requires much more computing resources than any of the other models analyzed. Training these CNN's–which is a rather simple implementation considering these complexity that top-notch CNN can reach—took around 6 hours in the computer used for the experiments (Intel® Core™ i5-4300U CPU at 1.9GHz with 8GB RAM), while typical training time for fastText and XGBoost ranged from 3 to 8 minutes for the former and 0.2 to 2.2 hours for the latter. Nevertheless, from the initial results found for CNN, this model might be able to achieve a competitive level with more elaborate network configurations and given the necessary computing power.

5 CONCLUSION

Throughout this work, the application of distributed text representations combined with modern classification algorithms implementations to the HTC task has been analyzed. After an observant literature research and careful examination of related works, three noticeable word embeddings generation methods—GloVe, word2vec, and fastText—and three prominent classification models—fastText, XGBoost, and CNN—that recently improved the results for the typical text classification and could potentially provide similar advancements for the hierarchical specialization were identified. The possibility to exploit the hierarchical structure to build classification models using LCPN strategy, virtual categories, and ancestry was noticed.

In order to assess the feasibility and effectiveness of these representations, models, and strategies to the HTC task, experiments using the RCV1 dataset were performed. An evaluation of the models using flat and hierarchical measures confirmed that the former are inadequate for the HTC context. A strong correlation between hierarchical and LCA measures was also identified, that presumably occurs because the underlying class hierarchy of this dataset is rather shallow. Although the classification models created using hierarchical strategies reduce the final F_1 values in comparison to flat approaches, they yielded improvements either on precision or recall.

FastText was the outstanding method both as a classifier and as a word embedding generator. The algorithm seems to owe most of its superiority to the way it estimates class-oriented word embeddings in supervised mode. These findings support the increasing understanding that combining task-specific word embeddings provides the best results for text classification (KIM, 2014; TANG et al., 2014), to which its hierarchical specialization can be now included. A direct comparison between other methods and ours is not available because previous studies have neither used hierarchical measures to asses the effectiveness or have not used the RCV1 dataset nor used a single-labeled version of it. Nevertheless, the $\text{LCA}F_1$ of 0.871 is considered a remarkable achievement. Although the other classification models do not reach competitive results, they are still worth of further investigation as exploring their flexibility could still provide promising improvements.

5.1 Future Work

These methods can be applied to the PubMed data to check how they extend to the medical text context—in particular the usage of fastText. As BioASQ provides pre-trained word embeddings generated with word2vec using a considerable amount of medical texts, comparing them with those that fastText creates in supervised mode should provide us with evidence for a more general understanding on how their quality affects the final classification results. Besides that, as the Mesh hierarchy is much larger than RCV1's in all senses, it would be useful to confront the hierarchical and LCA measures in order to confirm the hypothesis about their correlation.

Although CNN is among the models that exhibited the worst effectiveness, it deserves further investigation as this initial impression contradicts the expectations set by other studies. At the same time, a deeper comprehension of its architecture is necessary to understand and apply it to the HTC context. Besides that, it is worthy to investigate how effective can LSTM's be with this problem, as their ability to handle sequential data matches the ordered nature of texts.

In the long term, there are plans to research on the training objective used for HTC problems. Softmax was used in all experiments reported in this work, as this was the most suitable multi-class function available in the algorithm implementations used for these experiments. However, both XGBoost and CNN (through the Keras API) allow for the loss function customization. Finding a differentiable function that approximates either hF_1 or LCAF_1 and using it as the loss function rather than the softmax could finally bring together state-of-the-art algorithms with hierarchical information to create a method that implements a global HTC approach.

REFERENCES

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval**. 2nd. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 9780321416919.

BALIKAS, G.; AMINI, M. An empirical study on large scale text classification with skip-gram embeddings. **CoRR**, abs/1606.06623, 2016.

BLEI, D. M. Probabilistic topic models. **Communications of the ACM**, ACM, v. 55, n. 4, p. 77–84, 2012.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent Dirichlet allocation. **Journal of Machine Learning Research**, v. 3, n. Jan, p. 993–1022, 2003.

CESA-BIANCHI, N.; GENTILE, C.; ZANIBONI, L. Hierarchical classification: combining Bayes with SVM. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 23., 2006. **Proceedings of...** [S.l.], 2006. p. 177–184.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 22., 2016, San Francisco, California. **Proceedings of...** [S.l.], 2016. p. 785–794.

CHOLLET, F. et al. **Keras**. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>.

COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 25., 2008. **Proceedings of...** [S.l.]: ACM, 2008. p. 160–167.

COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K.; KUKSA, P. Natural language processing (almost) from scratch. **Journal of Machine Learning Research**, v. 12, n. Aug, p. 2493–2537, 2011.

COSTA, E.; LORENA, A.; CARVALHO, A.; FREITAS, A. A review of performance evaluation measures for hierarchical classifiers. In: **Evaluation Methods for Machine Learning II**: papers from the aaai-2007 workshop. [S.l.: s.n.], 2007. p. 1–6.

CRAIN, S. P.; ZHOU, K.; YANG, S.-H.; ZHA, H. Dimensionality reduction and topic modeling: From latent semantic indexing to latent dirichlet allocation and beyond. In: AGGARWAL, C. C.; ZHAI, C. (Eds.). **Mining text data**. [S.l.]: Springer, 2012. p. 129–161.

DEERWESTER, S.; DUMAIS, S. T.; FURNAS, G. W.; LANDAUER, T. K.; HARSHMAN, R. Indexing by latent semantic analysis. **Journal of the American Society for Information Science**, American Documentation Institute, v. 41, n. 6, p. 391, 1990.

DUMAIS, S.; CHEN, H. Hierarchical classification of web content. In: ANNUAL INTERNATIONAL ASSOCIATION FOR COMPUTING MACHINERY'S SPECIAL INTEREST GROUP ON INFORMATION RETRIEVAL'S CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 23., 2000, Athens, Greece. **Proceedings of...** New York, NY, USA: ACM, 2000. (SIGIR '00), p. 256–263. ISBN 1-58113-226-3.

ECKSTEIN, L.; KUEHL, N.; SATZGER, G. Towards extracting customer needs from incident tickets in it services. In: IEEE CONFERENCE ON BUSINESS INFORMATICS (CBI), 18., 2016, Paris. **Proceedings of...** [S.l.], 2016. v. 1, p. 200–207.

ESULI, A.; FAGNI, T.; SEBASTIANI, F. Boosting multi-label hierarchical text categorization. **Information Retrieval**, Springer, v. 11, n. 4, p. 287–313, 2008.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FELDMAN, R.; SANGER, J. **The text mining handbook**: advanced approaches in analyzing unstructured data. [S.l.]: Cambridge university press, 2007.

FREEMAN, J. A.; SKAPURA, D. M. **Neural Networks**: Algorithms, applications, and programming techniques. USA: Addison-Wesley Publishing Company, 1991.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.

GOODMAN, J. Classes for fast maximum entropy training. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, 2001, Salt Lake City, EUA. **Proceedings of...** [S.l.], 2001. p. 561–564.

GOVERNATORI, G. Exploiting properties of legislative texts to improve classification accuracy. In: ANNUAL CONFERENCE ON LEGAL KNOWLEDGE AND INFORMATION SYSTEMS (JURIX), 22., 2009, Rotterdam, The Netherlands. **Proceedings of...** [S.l.]: IOS Press, 2009. v. 205, p. 136.

GRAOVAC, J.; KOVAČEVIĆ, J.; PAVLOVIĆ-LAŽETIĆ, G. Hierarchical vs. flat n-gram-based text categorization: Can we do better? **Computer Science and Information Systems**, v. 14, n. 1, p. 103–121, 2017.

GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. LSTM: A search space odyssey. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, 2016.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining**: Concepts and techniques. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791.

HERSH, W.; BUCKLEY, C.; LEONE, T.; HICKAM, D. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: ANNUAL INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 17., 1994, Dublin, Ireland. **Proceedings of...** [S.l.], 1994. p. 192–201.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HUANG, C.; QIU, X.; HUANG, X. Text classification with document embeddings. In: **Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data**. [S.l.]: Springer, 2014. p. 131–140.

INGERSOLL, G. S.; MORTON, T. S.; FARRIS, A. L. **Taming text**: how to find, organize, and manipulate it. [S.l.]: Manning Publications Co., 2013.

JOACHIMS, T. Text categorization with suport vector machines: Learning with many relevant features. In: EUROPEAN CONFERENCE ON MACHINE LEARNING, 10., 1998, Chemnitz, Alemanha. **Proceedings of...** [S.l.], 1998. p. 137–142.

JOHNSON, R.; ZHANG, T. Effective use of word order for text categorization with convolutional neural networks. In: HUMAN LANGUAGE TECHNOLOGIES: THE 2015 ANNUAL CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of...** [S.l.], 2015. p. 103—112.

JORDAN, M. I.; JACOBS, R. A. Hierarchical mixtures of experts and the em algorithm. **Neural Computation**, MIT Press, v. 6, n. 2, p. 181–214, 1994.

JOULIN, A.; GRAVE, E.; MIKOLOV, P. B. T. Bag of tricks for efficient text classification. In: CONFERENCE OF THE EUROPEAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 15., 2017, Valencia, Espanha. **Proceedings of...** [S.l.], 2017. v. 2, p. 427–431.

KIM, Y. Convolutional neural networks for sentence classification. In: 19TH CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2014, Doha, Qatar. **Proceedings of...** [S.l.], 2014. p. 1746–1751.

KIRITCHENKO, S.; MATWIN, S.; NOCK, R.; FAMILI, A. F. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In: CONFERENCE OF THE CANADIAN SOCIETY FOR COMPUTATIONAL STUDIES OF INTELLIGENCE, 19., 2006, Quebec City, Quebec, Canada. **Proceedings of...**: Advances in artificial intelligence. [S.l.], 2006. p. 395–406.

KOLLER, D.; SAHAMI, M. Hierarchically classifying documents using very few words. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 14., 1997, Nashville, TN, USA. **Proceedings of...** San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. p. 170–178. ISBN 1-55860-486-3.

KOSMOPOULOS, A.; PARTALAS, I.; GAUSSIER, E.; PALIOURAS, G.; ANDROUTSOPOULOS, I. Evaluation measures for hierarchical classification: a unified view and novel approaches. **Data Mining and Knowledge Discovery**, Springer, v. 29, n. 3, p. 820–865, 2015.

KUSNER, M.; SUN, Y.; KOLKIN, N.; WEINBERGER, K. From word embeddings to document distances. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 32., 2015, Lille, France. **Proceedings of...** [S.l.], 2015. v. 37, p. 957–966.

LAI, S.; XU, L.; LIU, K.; ZHAO, J. Recurrent convolutional neural networks for text classification. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 29. **Proceedings of...** [S.l.], 2015. v. 333, p. 2267–2273.

LE, Q. V.; MIKOLOV, T. Distributed representations of sentences and documents. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 31., Beijing, China. **Proceedings of...** [S.l.]: PMLR, 2014. (Proceedings of Machine Learning Research, v. 32), p. 1188–1196.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Research, v. 521, n. 7553, p. 436–444, 2015.

LEWIS, D. D.; YANG, Y.; ROSE, T. G.; LI, F. Rcv1: A new benchmark collection for text categorization research. **Journal of Machine Learning Research**, v. 5, n. Apr, p. 361–397, 2004.

LI, Q.; SHAH, S.; LIU, X.; NOURBAKHSH, A.; FANG, R. Tweet topic classification using distributed language representations. In: IEEE/WIC/ACM INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE, 2016, Omaha, NE, USA. **Proceedings of...** [S.l.], 2016. p. 81–88.

MA, C.; XU, W.; LI, P.; YAN, Y. Distributional representations of words for short text classification. In: NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES. **Proceedings of...** [S.l.], 2015. p. 33–38.

MAAS, A. L.; DALY, R. E.; PHAM, P. T.; HUANG, D.; NG, A. Y.; POTTS, C. Learning word vectors for sentiment analysis. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 49., 2011. **Proceedings of...** Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (HLT '11), p. 142–150. ISBN 978-1-932432-87-9.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to information retrieval**. [S.l.]: Cambridge University Press Cambridge, 2008. v. 1.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **CoRR**, abs/1301.3781, 2013.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2013. p. 3111–3119.

NAKOV, P.; RITTER, A.; ROSENTHAL, S.; SEBASTIANI, F.; STOYANOV, V. Semeval-2016 task 4: Sentiment analysis in twitter. In: INTERNATIONAL WORKSHOP ON SEMANTIC EVALUATION (SEMEVAL), 10., 2016. **Proceedings of...** San Diego, California: Association for Computational Linguistics, 2016. p. 1–18.

NAKOV, P.; ROSENTHAL, S.; KOZAREVA, Z.; STOYANOV, V.; RITTER, A.; WILSON, T. Semeval-2013 task 2: Sentiment analysis in twitter. In: JOINT CONFERENCE ON LEXICAL AND COMPUTATIONAL SEMANTICS AND INTERNATIONAL WORKSHOP ON SEMANTIC EVALUATION, 2, 7., 2013. **Proceedings of...** Atlanta, Georgia, USA: Association for Computational Linguistics, 2013. v. 2, p. 312–320.

PANG, B.; LEE, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: ANNUAL MEETING ON ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 42. **Proceedings of...** [S.l.], 2004. p. 271.

PARTALAS, I.; KOSMOPOULOS, A.; BASKIOTIS, N.; ARTIÈRES, T.; PALIOURAS, G.; GAUSSIER, É.; ANDROUTSOPOULOS, I.; AMINI, M.; GALLINARI, P. LSHTC: A benchmark for large-scale text classification. **CoRR**, abs/1503.08581, 2015.

PENG, S.; YOU, R.; WANG, H.; ZHAI, C.; MAMITSUKA, H.; ZHU, S. Deepmesh: deep semantic representation for improving large-scale mesh indexing. **Bioinformatics**, Oxford University Press, v. 32, n. 12, p. i70–i79, 2016.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global vectors for word representation. In: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2014, Doha, Qatar. **Proceedings of...** [S.l.], 2014. v. 14, p. 1532–1543.

PORTER, M. F. An algorithm for suffix stripping. **Program**, MCB UP Ltd, v. 14, n. 3, p. 130–137, 1980.

PROVOST, F.; FAWCETT, T. **Data Science for Business**: What you need to know about data mining and data-analytic thinking. [S.l.]: O'Reilly Media, Inc., 2013.

PUURULA, A.; READ, J.; BIFET, A. Kaggle LSHTC4 winning solution. **CoRR**, abs/1405.0546, 2014.

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, Springer, v. 1, n. 1, p. 81–106, 1986.

RUIZ, M. E.; SRINIVASAN, P. Hierarchical text categorization using neural networks. **Information Retrieval**, Springer, v. 5, n. 1, p. 87–118, 2002.

SAHAMI, M. Learning limited dependence bayesian classifiers. In: 2ND INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (KDD) OF THE ASSOCIATION FOR THE ADVANCEMENT OF ARTIFICIAL INTELLIGENCE (AAAI). **Proceeding of...** [S.l.], 1996. v. 96, p. 335–338.

SCHAPIRE, R. E.; SINGER, Y. Improved boosting algorithms using confidence-rated predictions. In: ANNUAL CONFERENCE ON COMPUTATIONAL LEARNING THEORY, 11., 1998, Madison, WI, USA. **Proceedings of...** [S.l.], 1998. p. 80–91.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, Elsevier, v. 61, p. 85–117, 2015.

SCHÜTZE, H.; HULL, D. A.; PEDERSEN, J. O. A comparison of classifiers and document representations for the routing problem. In: ANNUAL INTERNATIONAL ASSOCIATION FOR COMPUTING MACHINERY'S SPECIAL INTEREST GROUP ON INFORMATION RETRIEVAL'S CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 18. **Proceedings of...** [S.l.], 1995. p. 229–237.

SCHWEIGHOFER, E.; RAUBER, A.; DITTENBACH, M. Automatic text representation, classification and labeling in european law. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND LAW, 8. **Proceedings of...** [S.l.], 2001. p. 78–87.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM Computing Surveys**, ACM, v. 34, n. 1, p. 1–47, 2002.

SILLA JR., C. N.; FREITAS, A. A. A survey of hierarchical classification across different application domains. **Data Mining and Knowledge Discovery**, Kluwer Academic Publishers, v. 22, n. 1-2, p. 31–72, 2011.

SOCHER, R.; PERELYGIN, A.; WU, J. Y.; CHUANG, J.; MANNING, C. D.; NG, A. Y.; POTTS, C. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2013, Seatle, USA. **Proceedings of...** [S.l.], 2013. v. 1631, p. 1642.

SULLIVAN, D. RIP DMOZ: The open directory project is closing. **Search Engine Land**, online (accessed 22-May-2017), 2017. Disponível em: <http://searchengineland.com/rip-dmoz-open-directory-project-closing-270291>. Acesso em: 22 maio 2017.

SUN, A.; LIM, E.-P. Hierarchical text classification and evaluation. In: IEEE INTERNATIONAL CONFERENCE ON DATA MINING, 2001, San Jose, CA, USA. **Proceedings of...** Washington, DC, USA: IEEE Computer Society, 2001. (ICDM '01), p. 521–528. ISBN 0-7695-1119-8.

TAI, K. S.; SOCHER, R.; MANNING, C. D. Improved semantic representations from tree-structured long short-term memory networks. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS AND INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING OF THE ASIAN FEDERATION OF NATURAL LANGUAGE PROCESSING, 53, 7., 2015, Beijing, China. **Proceedings of...** [S.l.], 2015. v. 1: long papers, p. 1556–1566.

TANG, D.; WEI, F.; YANG, N.; ZHOU, M.; LIU, T.; QIN, B. Learning sentiment-specific word embedding for twitter sentiment classification. In: 52ND ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 2014. **Proceedings of...** [S.l.], 2014. p. 1555–1565.

THOMEE, B.; SHAMMA, D. A.; FRIEDLAND, G.; ELIZALDE, B.; NI, K.; POLAND, D.; BORTH, D.; LI, L.-J. YFCC100M: The new data in multimedia research. **Communications of the ACM**, ACM, v. 59, n. 2, p. 64–73, 2016.

THOMPSON, P. Automatic categorization of case law. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND LAW, 8., 2001. **Proceedings of...** [S.l.], 2001. p. 70–77.

TRAPPEY, A. J.; HSU, F.-C.; TRAPPEY, C. V.; LIN, C.-I. Development of a patent document classification and search platform using a back-propagation network. **Expert Systems with Applications**, Elsevier, v. 31, n. 4, p. 755–765, 2006.

TRIESCHNIGG, D.; PEZIK, P.; LEE, V.; JONG, F. de; KRAAIJ, W.; REBHOLZ-SCHUHMANN, D. MeSH Up: effective mesh text classification for improved document retrieval. **Bioinformatics**, v. 25, n. 11, p. 1412–1418, 2009.

TSATSARONIS, G.; BALIKAS, G.; MALAKASIOTIS, P.; PARTALAS, I.; ZSCHUNKE, M.; ALVERS, M. R.; WEISSENBORN, D.; KRITHARA, A.; PETRIDIS, S.; POLYCHRONOPOULOS, D. et al. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. **BMC Bioinformatics**, BioMed Central, v. 16, n. 1, 2015.

VOSOUGHI, S.; VIJAYARAGHAVAN, P.; ROY, D. Tweet2vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder. In: 39TH INTERNATIONAL ASSOCIATION FOR COMPUTING MACHINERY'S SPECIAL INTEREST GROUP ON INFORMATION RETRIEVAL CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 39., 2016. **Proceedings of...** [S.l.], 2016. p. 1041–1044.

WESTON, J.; BENGIO, S.; USUNIER, N. WSABIE: Scaling up to large vocabulary image annotation. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 22., 2011, Barcelona, Espanha. **Proceedings of...** [S.l.], 2011. v. 11, p. 2764–2770.

WIENER, E.; PEDERSEN, J. O.; WEIGEND, A. S. et al. A neural network approach to topic spotting. In: 4TH ANNUAL SYMPOSIUM ON DOCUMENT ANALYSIS AND INFORMATION RETRIEVAL, 2015. **Proceedings of...** [S.l.], 1995. v. 317, p. 332.

YING, C. et al. Novel top-down methods for hierarchical text classification. **Procedia Engineering**, Elsevier, v. 24, p. 329–334, 2011.

YU, B.; XU, Z.; LI, C. Latent semantic analysis for text categorization using neural network. **Knowledge-Based Systems**, Elsevier, v. 21, n. 8, p. 900–904, 2008.

ZENG, C.; LI, T.; SHWARTZ, L.; GRABARNIK, G. Y. Hierarchical multi-label classification over ticket data using contextual loss. In: NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (NOMS), IEEE. **Proceedings of...** [S.l.], 2014. p. 1–8.

ZHANG, M.-L.; ZHOU, Z.-H. Multilabel neural networks with applications to functional genomics and text categorization. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 18, n. 10, p. 1338–1351, 2006.

ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level convolutional networks for text classification. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 28., 2015. **Proceedings of...** [S.l.], 2015. p. 649–657.

ZIPF, G. K. **The Psycho-Biology of Language**. [S.l.]: Houghton, Mifflin, 1935.