Received 9 January 2024, accepted 1 February 2024, date of publication 19 February 2024, date of current version 23 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3367246

RESEARCH ARTICLE

A Comparative Analysis of Word Embeddings Techniques for Italian News Categorization

FEDERICA ROLLO, GIOVANNI BONISOLI, AND LAURA PO
Department of Engineering "Enzo Ferrari," University of Modena and Reggio Emilia, 41121 Modena, Italy

Corresponding author: Federica Rollo (federica.rollo@unimore.it)

This work was supported in part by the project "Deep Learning for Urban Event Extraction from News and Social Media Streams" funded by the Department of Engineering "Enzo Ferrari," University of Modena and Reggio Emilia.

ABSTRACT Text categorization remains a formidable challenge in information retrieval, requiring effective strategies, especially when applied to low-resource languages such as Italian. This paper delves into the intricacies of categorizing Italian news articles, addressing the complexities arising from the language's unique structure and writing style. The implemented methodology involves preprocessing the text, generating word embeddings, conducting feature engineering to extract meaningful representations, and training a classifier using the document vectors. The evaluation of the model's performance is done on a partitioned dataset with a training set for model training and a test set for categorization, allowing assessment of its efficacy on unseen data. Within this paper, we assessed fifteen classifiers for the categorization of Italian news articles, scrutinizing eight models and three approaches for combining word embeddings to derive document vectors. We conducted a comparative analysis between established models such as Word2Vec and FastText and six novel Italian models pre-trained on native datasets. A significant highlight of our work is the introduction of an Italian GloVe model, previously absent for the Italian language. The datasets selected for testing the models' performances are DICE, a dataset of 10,395 crime news articles extracted from an Italian newspaper, and RCV2-it, a collection of 28,405 Italian news stories released by the multinational media company Reuters Ltd. The tests conducted achieved as the best F-scores 84% and 93%. The results underscore the efficacy of the Support Vector Classification algorithm, while also revealing the inefficacy of Gaussian Naive Bayes, Bernoulli Naive Bayes, and Decision Tree models within the domain of text categorization. The comparison of the word embedding models revealed the better performance of Word2Vec and GloVe concerning FastText. The broader impact of this paper lies not only in advancing text categorization methodologies for Italian documents but also in enriching the linguistic landscape by releasing six novel Italian word embedding models.

INDEX TERMS Text categorization, text classification, word embeddings, natural language processing (NLP), Word2Vec, FastText, GloVe.

I. INTRODUCTION

With the exponential growth of digital content across various domains, the task of effectively organizing and managing vast amounts of textual data has become increasingly vital. Text categorization (sometimes referred to as *text classification*) facilitates efficient data organization, retrieval, and analysis and enables plenty of further tasks in the field of Natural Language Processing (NLP), such as automated information retrieval, sentiment analysis, spam detection. Given a predefined list of labels (also known as categories or classes), text categorization aims at assigning one or more labels to documents based on their content. This process entails analyzing the inherent characteristics of the text, extracting relevant features, and employing Machine Learning algorithms to classify documents into predefined categories. By automating this process, text categorization offers a powerful solution to deal with the overwhelming volume of textual data, facilitates decision-making processes and streamlines information management.

The associate editor coordinating the review of this manuscript and approving it for publication was Maria Chiara Caschera.

Focusing on the specific use case of news articles, categorization poses unique challenges due to the inherent structural complexities of news. Unlike other types of text, news articles follow specific conventions and formats that can impact the effectiveness of categorization algorithms. The structure of news articles is known as an *inverted pyramid* and consists of introducing the main information at the beginning, providing details in the middle and pushing general and background elements to the margin [1], [2]. The main challenges for news categorization arise from the presence of noise and irrelevant information within the articles, the use of diverse writing styles, and the varying lengths (from brief news snippets to lengthy investigative reports). Noise and irrelevant information within news articles can lead to misclassification because they make it difficult to distinguish between important and less critical information. Filtering out the noise and identifying the most relevant portions of the article can improve the accuracy of news article categorization. Besides, each newspaper has its own editorial guidelines, writing conventions, and biases and news articles can reflect the inverted pyramid more or less accurately. Variations in writing styles can impact the consistency and accuracy of categorization algorithms. A model trained on articles from one newspaper may not generalize well to articles from another source due to differences in terminology, language usage, and overall tone. Adapting categorization algorithms to handle these variations and account for different writing styles is crucial for the accurate classification of news articles.

Several approaches to text categorization exploit word embeddings to obtain the document vector representations to embed into Machine Learning algorithms. Word embedding is a continuous vector representation of words that encodes the meaning of the word, such that the words that are closer in the vector space are expected to be similar in meaning. There are different models that can be used to derive these vectors, such as Word2Vec [3], Glove [4], FastText [5].

This research paper aims to conduct a comprehensive investigation into the application of word embeddings for categorizing Italian news articles. While numerous comparative studies exist for languages like English, there is a notable scarcity of tests involving Word2Vec and FastText within the context of Italian language text categorization, with GloVe conspicuously absent.

To the best of our knowledge, there is no prior work that compares different word embedding techniques for the categorization of Italian news articles. Consequently, our research serves as a potential benchmark for the scientific community, making a distinctive contribution by addressing this gap and providing valuable insights into the effective application of word embeddings in the categorization of Italian documents.

The paper's primary contributions are outlined as follows:

- **Release of new Italian Word Embedding Models:** The paper makes a significant contribution by providing six new Italian word embedding models. Specifically, it contributes two Word2Vec models, two GloVe models, and two FastText models. These models have been made available as open-source software on Zenodo repositories [6], [7], [8], facilitating their potential retraining on other datasets. As a result of this work, Italian GloVe models, previously unavailable, now stand as state-of-the-art models.
- **Comparative Evaluation of Italian Word Embedding Models in Word Analogy Tasks**: The paper conducts an initial assessment of word embedding models to gauge their performance in resolving word analogies within the Italian language. This evaluation is discussed in Section IV-C. This analysis provides insights into the strengths and weaknesses of different word embedding models in capturing semantic relationships within the Italian language, filling a gap in the existing literature.
- **Comparison of Supervised Machine Learning Algorithms for Document Categorization:** In Section V, the paper carries out extensive experiments that involve comparing fifteen supervised machine learning algorithms. These experiments are conducted on two datasets: one comprises 1,118 manually annotated news articles categorized into 6 distinct groups, and the other contains 6,478 documents categorized into 4 groups. The objectives of these experiments are as follows:
    - Evaluation of document vectors based on TF-IDF and word embeddings (Section V-A).
    - Exploration of various techniques for combining word embeddings to generate document vectors (Section V-B).
    - Analysis of different training configurations for Word2Vec, GloVe, and FastText models (Section V-C).
    - Assessment of the impact of utilizing different training datasets for Word2Vec and FastText models (Section V-D).

  This comprehensive comparison offers valuable insights into the effectiveness of different approaches to feature engineering and word embedding models in the context of document classification tasks for the Italian language. The results that emerged in the experimental evaluation can be leveraged by the scientific community to make informed decisions when implementing machine learning models for categorizing Italian news articles, ultimately advancing the state-of-the-art in the field.

This paper extends our previous research works [9], [10] where we limited to test only one word embedding model, Word2Vec, to obtain the document vector representations, and some supervised and unsupervised algorithms to categorize Italian crime news articles. With respect to the previous works, we extended the comparison to FastText and GloVe, trained new word embedding models, refined one labelled dataset by manually annotating a portion of the whole dataset, added one dataset to tests, and excluded the unsupervised approaches.

The paper is organized as follows: Section II delve into a comprehensive exploration of the current state of research in text categorization focusing on word embedding techniques, while Section III presents our methodology to obtain the document vector representations and categorize the documents. In Section IV, we describe the set-up of our experiments that are evaluated in Section V. Section VII is devoted to discussion and conclusions.

II. LITERATURE REVIEW

This section delves into a comprehensive exploration of the current state of research in text categorization. We start by tracing the evolution of methodologies, beginning with traditional approaches and progressing to advanced Machine Learning and Deep Learning models. We depict the ascendancy of neural network-based Natural Language Processing methods, including CNNs, RNNs, LSTM, and Transformers. We recognize the pivotal role played by word embeddings in text representation and their significant impact on enhancing NLP tasks, such as text categorization. We refer to numerous studies that have compared the performance of various word embedding models, highlighting their effectiveness in capturing word similarity and classifying documents. Notably, we emphasize a research gap in the context of Italian language text categorization. While comparative studies involving Word2Vec, GloVe, and FastText have been conducted for other languages, such evaluations within the Italian language remain largely unexplored. This highlights the importance of this work and its contribution to future research.

Over the years, researchers have developed and explored various approaches and techniques for text categorization, ranging from traditional methods to more advanced Machine Learning and Deep Learning models, as reported in recent surveys [11], [12], [13].

Traditional methods refer to approaches that require an initial preprocessing stage to convert unprocessed textual input into structured features suitable for input to a Machine Learning model. More recently, there has been a notable surge in the adoption of neural network-based NLP methods using deep learning architectures like CNNs [14], [15], RNNs [16], [17], LSTM [18], [19], [20], and the cutting-edge Transformers [21], [22], [23]. In contrast to conventional methodologies, these approaches obviate the necessity for human-designed rules and features by autonomously producing semantically significant representations and encapsulating intricate and non-linear correlations within input data. Nevertheless, these benefits are counterbalanced by augmented intricacy and heightened computational requirements.

Most of the works in the literature focus on the traditional methods previously described. Feature extraction techniques commonly employed include keyword extraction, term frequency, term frequency-inverse document frequency (TF-IDF), N-grams, Bag-of-words, and word embeddings. These methods are often combined with conventional Machine Learning models for classification purposes, including Naive Bayes, Decision Tree, Support Vector Machine, and K-Nearest Neighbor [24], [25], [26], [27]. Among the options for feature extraction, word embedding stands out as a contemporary text representation method that is rapidly gaining widespread usage. Using word embeddings to represent features extracted from the text improves the performance in many NLP tasks, including text categorization [27], [28], [29], [30], [31], [32], [33], [34], [35], [36].

In several works, the performances of different word embedding models have been compared in capturing word similarity and categorizing documents. Chandrasekaran and Mago [37] analyzed the sensitivity of various word embedding models with respect to the complexity of the sentences and demonstrated that sentence complexity has a significant impact on the performance of the embedding models, however, models based on BERT [21] perform better than Word2Vec and GloVe. The authors of [27] demonstrated that weighting Word2Vec-based word embeddings by TF-IDF can outperform the use of simple word embeddings for text categorization using a linear Support Vector Classification (SVC). Averaging word embeddings using TF-IDF weights is a powerful technique that can leverage the contextual information and the importance of words within a document, as supported also in [38] where the model used is GloVe. In [39], the effect of stemming strategies on text categorization of Arabic documents has been analyzed by comparing the performances of different deep learning algorithms.

Word2Vec, GloVe and FastText have been compared in several works. Their performances depend on plenty of factors: the type of documents used (e.g., news articles, tweets, user comments), the language, the complexity of the text, how the training phase was implemented. Nugent et al. [40] classified English news articles according to seven categories of natural disasters (e.g., floods, fires, storms) using RF, SVM, CNN and HAN. The best performances were reported by FastText regardless of the classifier used. However, frequently, NLP tasks on English texts tend to perform better compared to the same tasks on other languages since a larger amount of data is available for the English language and models can derive advantages from extensive training. In [41], the authors proposed a binary sentiment classification of Arabic tweets using simple average and TF-IDF average of word embeddings. The best values of the F-score are obtained by applying NuSVC to FastText embeddings. However, tweets are very different from news articles because they are limited to 140 characters and use a simple sentence structure. Hate detection is another sub-task of text categorization; in [42] different algorithms are tested on Arabic content, when using Word2Vec the best performances are achieved by Linear SVC, and by Multilayer Perceptron when the embeddings are extracted by FastText. In [43], Word2Vec, Glove, and TF-IDF are tested for Named Entity Recognition on code-mixed data in English and Indian and the best result is achieved by Linear SVC with TF-IDF embeddings. In [44], the authors evaluate different methods on Portuguese intrinsically on syntactic and semantic analogies and extrinsically on POS tagging and sentence semantic similarity tasks. The best results are from Wang2Vec [45], a modification of Word2Vec made in order to take into account the lack of word order.

To the best of our knowledge, such a comparison between Word2Vec, GloVe and FastText for the categorization of Italian documents has not yet been performed. An analysis of the Italian word embeddings on the word analogy test has been conducted by Tripodi and Pira [46]. However, the evaluation involved only Word2Vec Skip-Gram and Word2Vec Continuous Bag-of-Words because only those Italian word embedding models were available in 2017. Another comparison on the word analogy test has been conducted by Berardi et al. [47] but involved only Word2Vec and Glove. Furthermore, we must keep in mind that previous works highlighted the absence of correlation between the performance of word vectors on word similarity and extrinsic evaluation on downstream tasks like text classification, NER, parsing and other [44], [48]. This reinforces the need for specific tests on text categorization. BERT-based embeddings have been tested with CNN and Logistic Regression (LR) to detect misogyny and aggressiveness on Italian tweets reaching an F1-score of 0.96 for misogyny and 0.85 for aggressiveness [49]. No comparison was made with respect to other word embedding models. Gambino and Pirrone demonstrated that the context-free embeddings of GloVe coupled with a deep neural classifier outperform the contextualized embeddings of a multilingual distribution of BERT for Italian sentiment analysis [50].

III. METHODOLOGY

Our implemented methodology, as depicted in Figure 1, encompasses several crucial phases. The initial step involves *pre-processing* the text extracted from the documents. During this phase, we tokenize the text and eliminate stop words, resulting in a curated list of relevant words for each document.

Subsequently, this curated word list serves as input for a model designed to generate word embeddings. These embeddings are aggregated using the TF-IDF weighted average approach, enabling us to calculate a distinct vector associated with each document. This phase, known as *feature engineering*, plays a pivotal role in extracting meaningful representations from the textual data. We have opted for the TF-IDF (Term Frequency-Inverse Document Frequency) function in this process. This choice is supported by prior research findings [27], [38] demonstrating its superior performance compared to a simple average.

Following the feature engineering phase, a classifier can be trained using the document vectors. This classifier is then equipped to categorize the input documents, thus facilitating the *categorization* phase of our methodology.

To evaluate the effectiveness of our approach, we partitioned the dataset into two subsets: the training set and the test set. The training set serves as the foundation for training a Machine Learning model, which is subsequently applied to categorize the documents within the test set. This division of data enables us to assess the model's performance on unseen data, a crucial step in gauging its efficacy.

A. FEATURE ENGINEERING

The process of feature engineering aims at converting text into feature vectors. The use of dense vectors to convey the meaning of words is known as *word embedding*. Word vectors are able to capture syntactic and semantic regularities of natural language for a more efficient understanding of the text. Among the different models for the distributed language representation, we selected for comparison three of them: Word2Vec, GloVe and FastText.

1) Word2Vec

In 2013, researchers from Google proposed Word2Vec [3] which is based on a shallow neural network with a single hidden layer. The input data of the network are generated by a window sliding on the text of the training corpus. Within this window, a target word is selected and the other words constitute the context. Two architectures can be used to compute the continuous vectors: the Skip-Gram (SG) and Continuous Bag-Of-Words (CBOW). The former aims to predict the context words given a target word, maximizing the probability of the context words given the target word. While, in the CBOW architecture, the model predicts the target word based on the context of surrounding words. The objective is to maximize the probability of the target word given its context. Through these "fake tasks", internal parameters of the network are adjusted to optimize the likelihood of correctly predicting the target word or context words. This allows for calculating the word embeddings, which is the real objective of training.

2) GloVe

Glove [4], developed by researchers from Stanford University in 2014, is based on the idea that word embeddings should encode not only the local context information of a word but also the global statistics of word co-occurrence in a corpus. Thus, the bilinear regression model used by GloVe combines two methods, i.e., the global matrix factorization and the local context window. The training process involves constructing a word co-occurrence matrix, which captures the frequency of how often words appear together in the same context within a given window size. This matrix is then used to compute the word embeddings by optimizing an objective function that seeks to minimize the difference between the dot product of two word embeddings and the logarithm of their co-occurrence probability. The model iteratively adjusts the embeddings to better approximate the observed co-occurrence probabilities. GloVe vectors capture the semantics of the target word with respect to its neighboring words.

3) FASTTEXT

FastText [5] can be considered an extension of Word2Vec. It was developed by Facebook AI Research in 2016 and

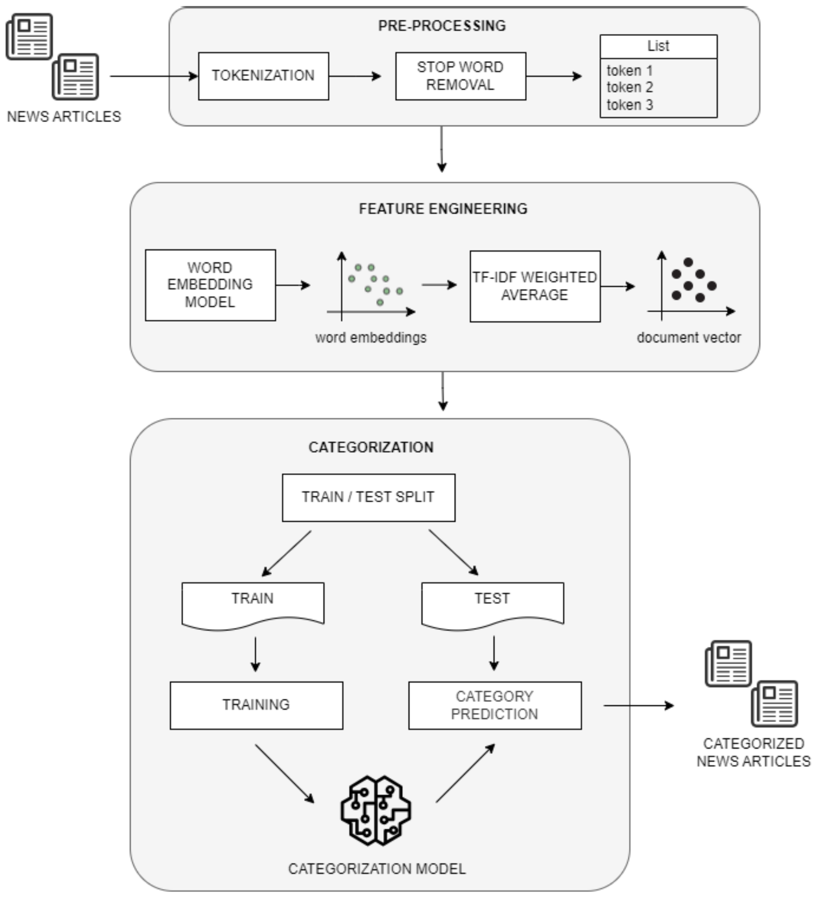


FIGURE 1. **Methodology.**

can be based on SG or the CBOW. The key difference between FastText and Word2Vec is the use of n-grams because while Word2Vec learns vectors only for complete words found in the training corpus, FastText learns vectors for the n-grams that are found within each word, as well as each complete word. Each word can be represented as a bag of character n-grams. The model learns the vector representations of these n-grams, considering their frequency and context information, and words are represented as the sum of these representations. In this way, FastText is able to capture morphological variations, prefixes, and suffixes, that contribute to the overall meaning of a word. The approach used by FastText allows for computing representations even of words that do not appear in the training data (vocabulary) and this is very important when dealing with rare words and languages with rich morphology.

B. CATEGORIZATION

Once the document vectors of each news article in the dataset are extracted, plenty of algorithms can be used to identify the category each news article belongs to. Both supervised and unsupervised techniques can be taken into account. The *supervised text categorization* algorithms predict the topic of a document within a predefined set of categories, named labels. This approach relies on labelled training data, where each text document is associated with a known category. During the training of the categorization model, the model can capture underlying patterns and relationships between the word embeddings and the target categories. This learned knowledge can then be applied to accurately classify new, unseen texts. However, the success of supervised text categorization heavily depends on the quality and representativeness of the training data. Adequate training data should cover various categories and account for the inherent diversity within the text corpus. Insufficient or biased training data may lead to poor generalization and inaccurate categorization of new texts.

Numerous supervised algorithms can be employed for text categorization. Most traditional methods include Naïve Bayes [51], [52], [53], [54], [55], k-nearest neighbors [56], [57], [58] and Support Vector Machines.

Naïve Bayes (NB) classifier is a probabilistic classifier. Given a document d and a set of classes $C = \{c_1, c_2, \ldots, c_k\}$, the algorithm estimates a posteriori probability $P(c|d)$ based on Bayes' theorem

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{1}$$

TABLE 1. Datasets description.

	MT	manualDICE	RCV2
Number of categories	6	6	4
Total # documents	5,510	1,118	21,593
# unique words	33,988	25,871	43,216
Total # words	1,186,807	458,274	2,324,909
Mean # words/doc	215	287	108
Total # sentences	43,355	21,539	130,905
Mean # sentences/doc	8	13	6
Mean # words/sentences	27	21	18

with strong independence assumptions between the features. There are several versions of this classifier that differ mainly by their assumptions about the distribution $P(d|c)$.

The *K-nearest neighbors* algorithm (KNN) is a non-parametric learning method that classifies data from a simple plurality vote of the k nearest neighbors in the training set. Each new point is assigned the class which has the most representatives within the k nearest neighbors of the point.

Other popular algorithms for text categorization algorithms are the ensemble methods, which use multiple classifiers to obtain better predictive performance than one single classifier alone. The most famous are Decision Tree, Random Forest, Bagging and Boosting techniques.

Decision tree [59] classifier is successfully used in many languages for text categorization [60], [61], [62]. The main idea is to create a decision tree based on the features of data points. The tree is built recursively by selecting the best attribute for each node and splitting the dataset into smaller subsets. The selection of attributes is done through Attribute Selection Measures, such as Information Gain or Gini Impurity [63]. The process stops when no more attributes or training instances remain.

Bagging classifier is based on the technique of Bootstrap Aggregating (Bagging), which splits the original dataset into k random subsets. On each subset, a base classifier is trained. Each new data point is submitted to all the k classifiers and the resulting predictions are aggregated (by major voting or averaging) into a final one. This procedure generally leads to better performance because the base classifiers are not correlated and this decreases the variance without increasing the bias [64].

Random Forest [65] classifier is an extension of the Bagging algorithm with Decision Tree as the base classifier. Random Forest applies the bagging also to the features. This process, sometimes called "feature bagging", is performed to handle the possible scenario where one or few features are more decisive than others in prediction. These features would be selected in many of the base classifiers, causing them to become correlated.

Extra Tree [66] classifier is a variant of Random Forest with two main differences: first, each tree is trained using the whole training sample, and second, the top-down splitting in base tree learners is randomized.

Boosting [67] technique involves building a sequence of models, where each subsequent model tries to correct the errors made by the previous model. The most famous algorithm based on boosting is *Adaboost* [67], where each subsequent model tries to focus on the samples that were misclassified by the previous model. At each iteration of the training process, a weight is assigned to each sample in the training set equal to the current error on that sample. This weight is higher for misclassified samples and lower for the other ones. This allows the new model to focus more on those samples which are wrongly classified. *Gradient Boosting* [68] is another widely used boosting algorithm, each subsequent model tries to correct the previous model by minimizing its error, through the computation of the gradient of the loss function. *XGBoost* [69] is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.

Support Vector Machine (SVM) [70] is another popular technique which employs a discriminative classifier for document categorization [71], [72], [73], [74]. Given a set of data points in a space, the idea behind SVM is building a hyperplane or set of hyperplanes to separate points not belonging to the same class. A good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class. The larger the margin, the lower the generalization error of the classifier, which means a low probability of overfitting. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function selected to suit the problem. The most classical kernel functions are linear, polynomial, radial basis function (rbf) and sigmoid.

Perceptron [75] is a linear classifier that makes its predictions based on a linear predictor function that combines a set of weights with the feature vector. In binary classification, if the dot product between the vector of the weights and the feature vector summed with a constant value (bias) is positive, the prediction is positive, otherwise is negative.

IV. EXPERIMENTAL SETUP

We conducted several experiments on two datasets to compare the performances of Word2Vec, GloVe and FastText in the categorization of Italian documents. We used different versions of the word embedding models to obtain the document representations that are then used to train fifteen machine learning classifiers.

In the following, we will describe the datasets used for the tests, the feature engineering (i.e., the process of converting words into word embeddings for each of the employed models and datasets), and the test to evaluate the ability of word embedding models to solve word analogies.

A. DATASETS

DICE [76] is a Dataset of Italian Crime News articles that we extracted from the newspaper Gazzetta di Modena[1]

[1]https://www.gazzettadimodena.it/

TABLE 2. Crime categories in MT and manualDICE datasets.

Category	MT	manualDICE	Total	% of test
Theft	2,359	559	2,918	19%
Drug	1,536	105	1,641	6%
Robbery	609	126	735	17%
Assault	421	111	532	21%
Fraud	404	108	512	21%
Murder	181	109	290	38%
Total	5,510	1,118	6,628	17%

TABLE 3. Category distribution in the RCV2 dataset.

Category	Train set	Test set	Total	% of test
MCAT	5,560	2,346	7,906	30%
CCAT	5,393	2,359	7,752	30%
ECAT	2,735	1,154	3,889	30%
GCAT	1,427	619	2,046	30%
Total	15,115	6,478	21,593	30%

which publishes daily news of events of the Modena province, in Italy. We extracted these news articles from the newspaper website by using a keyword search approach. Firstly, we selected a list of crime categories we were interested in and then we searched for each one of these crime categories on the website. In this way, it is supposed that the news articles retrieved are related to that crime category. Thus, the category used in the keyword search is used to label the news article, we refer to these labels as *newspaper categorization*. We recently released `DICE` under the CC BY-NC-SA 4.0 license.[2] The dataset contains 10,395 crime news articles of 13 crime categories and was created to address the lack of resources available for training NLP models, particularly for Italian event extraction and question answering systems. After some analyses of the dataset, we discovered that the newspaper categorization contains some errors. A subset of the 1,118 news articles of `DICE` has undergone manual annotation using a complex annotation schema which reveals, among other things, the correct category of the news. We indicate this dataset with the name `manualDICE`. On this dataset, the performance of newspaper categorization reaches 70% of precision, 63% of recall and 59% of F1-score.

In this work, we use `manualDICE` to test the categorization methodology. To train the machine learning algorithms for categorization we extracted 5,510 crime news articles from another newspaper named ModenaToday (`MT`); these news articles are classified by the editor of the newspaper itself according to the type of crime event described.

We selected another dataset for experiments, named `RCV2-it`. This dataset is obtained from the Reuters `RCV2` dataset,[3] a multilingual corpus that contains over 487,000 news stories from Reuters in thirteen languages, including Italian. All these news stories are labeled according to the multi-label and hierarchical schema of the `RCV1` dataset [77]. There are four main categories: MCAT, CCAT, ECAT and GCAT. `RCV2-it` is obtained from the Italian subset (28,405 items) of `RCV2`. We filtered out those items that fall into more than one of the four categories. With a view to making the text simpler, from now on, we will refer to the portion of `RCV2-it` we used as `RCV2`.

In Table 1, we reported some statistics on `MT`, `manualDICE` and `RCV2`. Table 2 shows the number of news articles for each category in `MT` and `manualDICE`. The percentage of news articles in `manualDICE` with respect to the whole dataset composed of `manualDICE` and `MT` for each category is variable, from 6% for "drug" to 38% for "murder". The distribution of categories in `RCV2` is reported in Table 3. In this case, we automatically split the dataset into balanced train and test sets. Therefore, the percentage of news articles in the test set with respect to the whole dataset is always the same, i.e., 30%.

B. WORD EMBEDDING MODELS

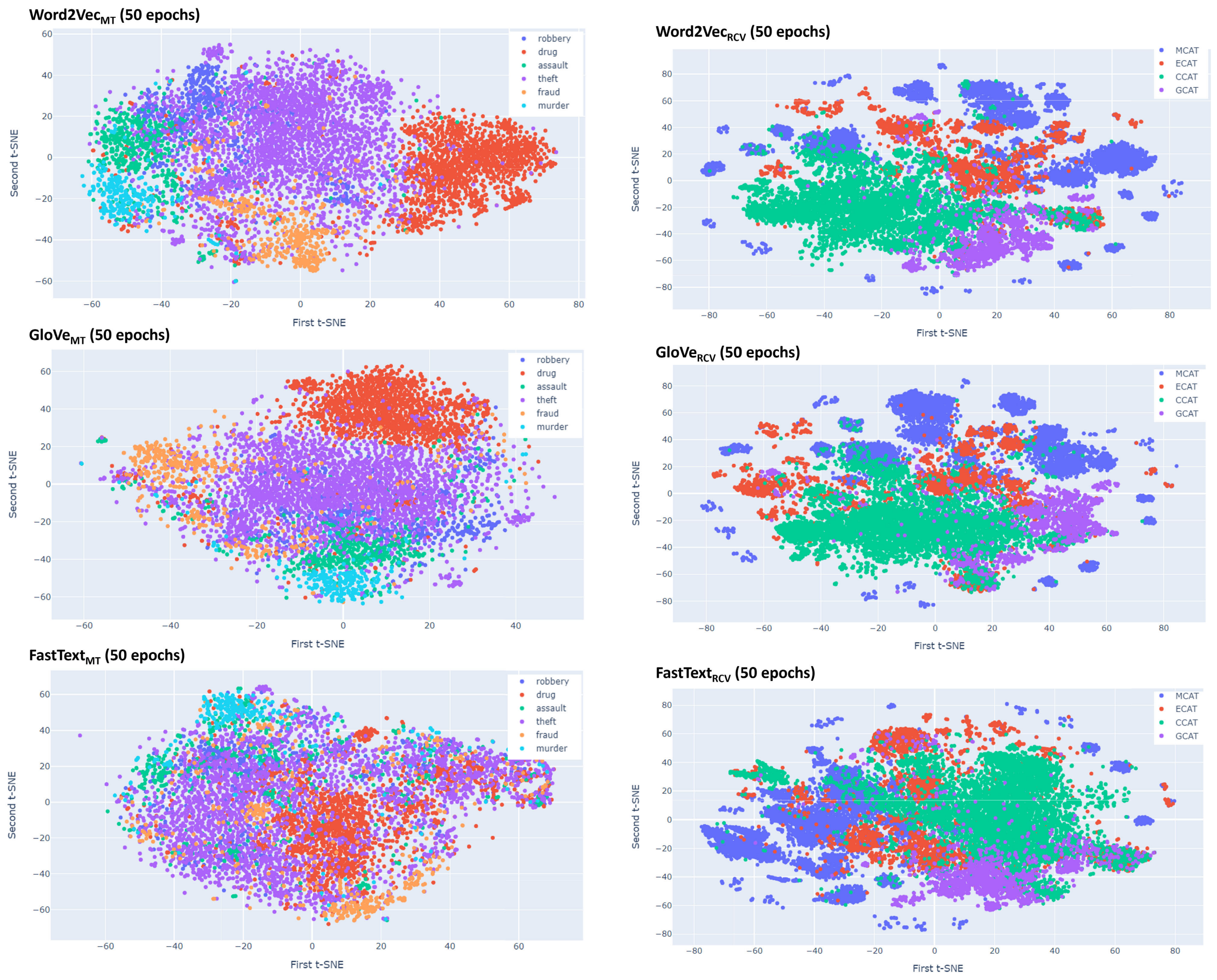
Document vectors for the news articles were computed using various word embedding models as detailed in Section III-A. To begin with, we looked for existing Italian models. We employed existing Word2Vec and FastText models, as there were no pre-trained Italian GloVe models available at the time of this study. These two models are referred to as the state-of-the-art (SOTA) models:

- **Word2Vec$_{\text{SOTA}}$**: This model is a Skip-gram Word2Vec model [78] pre-trained with 300-dimensional embeddings. The training data for this model included a dump of Wikipedia (as of April 1, 2019), main categories of Italian Google News, and anonymized chats between users and the customer care chatbot Laila.[4] The dataset used for training contained 17,305,401 sentences and 421,829,960 words, totaling 2.6 GB of raw text. The training parameters included a window size of 10 and 20 as the number of negative samples.
- **FastText$_{\text{SOTA}}$**: These are pre-trained Italian word vectors distributed by Facebook [79]. They were trained on Common Crawl (36,237,951,419 tokens) and Wikipedia (702,638,442 tokens) data using Continuous Bag of Words (CBOW) with position-weights, featuring 300 dimensions, character n-grams of length 5, a window size of 5, and 10 negatives. The exact number of training epochs is not specified.

In addition to these existing models, we trained Word2Vec, FastText, and GloVe from scratch using three distinct datasets:

- A dump of Italian Wikipedia (as of December 15, 2022), comprising 25,548,651 sentences and 526,640,982 words (3.2 GB of raw text).
- A dataset of Italian news (159,226 documents) from the webz.io platform, crawled in October 2015, containing 44,041,823 sentences and 44,544,385 words (244 MB).

[2]https://github.com/federicarollo/Italian-Crime-News
[3]https://trec.nist.gov/data/reuters/reuters.html
[4]https://www.laila.tech/



(a) manualDICE

(b) RCV2

FIGURE 2. **T-SNE visualization of the document vectors.**

- Either the MT dataset or the train set of RCV2, depending on whether the model was intended for generating embeddings for manualDICE or the test set of RCV2.

Word2Vec and FastText models are trained by using Python library gensim,[5] while Glove models are trained using the original library released by Stanford University.[6]

This resulted in the following released models:

- **Word2Vec$_{MT}$**: Word2Vec model trained on a dataset consisting of Italian Wikipedia, webz data, and MT;
- **Word2Vec$_{RCV}$**: Word2Vec model trained on a dataset consisting of Italian Wikipedia, webz data, and RCV2 train set;
- **FastText$_{MT}$**: FastText model trained on a dataset composed of Italian Wikipedia, webz data, and MT;
- **FastText$_{RCV}$**: FastText model trained on a dataset consisting of Italian Wikipedia, webz data, and RCV2 train set;
- **GloVe$_{MT}$**: GloVe model trained on a dataset consisting of Italian Wikipedia, webz data, and MT;
- **GloVe$_{RCV}$**: GloVe model trained on a dataset consisting of Italian Wikipedia, webz data, and RCV2 train set.

Each of these models underwent two training configurations, differing in the number of epochs: 20 epochs for the first configuration and 50 epochs for the second. Word2Vec models were trained with the same settings as Word2Vec$_{SOTA}$, while FastText models were trained similarly to FastText$_{SOTA}$. Glove models used a window size of 5, a vector size of 300, and a minimum word frequency of 20 during training.

To provide a visual representation of the proposed models, we employed Word2Vec$_{MT}$, GloVe$_{MT}$, and FastText$_{MT}$, each trained for 50 epochs, to compute document vectors for

[5]https://radimrehurek.com/gensim/models/keyedvectors.html
[6]https://github.com/stanfordnlp/GloVe

TABLE 4. Accuracy results on the word analogy test.

	#analogies	Word2Vec			GloVe		FastText		
		SOTA	MT	RCV2	MT	RCV2	SOTA	MT	RCV2
capital-common-countries	506	89.53%	84.19%	85.77%	**97.43%**	94.07%	95.45%	26.28%	27.47%
capital-world	4353	76.87%	75.60%	76.16%	77.33%	75.46%	**86.43%**	8.51%	8.77%
currency	866	7.05%	4.21%	4.33%	0.87%	0.74%	**7.55%**	1.11%	1.24%
city-in-state	2467	42.48%	29.35%	29.39%	42.81%	39.24%	**57.18%**	4.09%	4.58%
regione-capoluogo	342	47.37%	38.89%	36.84%	48.83%	47.95%	**59.65%**	12.87%	14.04%
family	381	75.85%	66.66%	69.55%	65.35%	70.08%	**77.69%**	37.27%	42.78%
accuracy on semantic	8915	60.11%	54.56%	55.01%	59.89%	57.95%	**69.78%**	8.97%	9.60%
adjective-to-adverb	930	14.84%	14.37%	15.86%	8.28%	9.43%	32.8%	70.69%	**71.26%**
opposite	552	18.18%	13.85%	12.34%	9.74%	9.31%	41.7%	46.97%	**48.27%**
comparative	12	0%	**16.66%**	8.33%	8.33%	8.33%	8.33%	0%	0%
superlative (absolute)	814	20.10%	18.90%	19.40%	11.00%	11.85%	76.38%	81.10%	**82.33%**
present-participle (gerund)	1056	63.54%	69.32%	68.47%	50.28%	50.76%	90.93%	**95.64%**	95.36%
nationality-adjective	1599	86.8%	85.93%	87.93%	**88.56%**	88.49%	85.74%	73.98%	74.48%
past-tense	1056	37.12%	41.29%	39.49%	27.37%	29.64%	69.51%	**76.52%**	**76.52%**
plural	1260	50.95%	48.57%	49.93%	38.57%	38.41%	**70.56%**	33.65%	34.05%
plural-verbs (3rd p.)	813	85.73%	83.64%	81.43%	57.44%	61.87%	93.6%	**97.17%**	96.92%
plural-verbs (1st p.)	930	0%	10.48%	10.48%	3.33%	3.33%	70.83%	**89.05%**	86.19%
present-remote-past-verbs	930	5.45%	13.33%	14.44%	0%	0%	**37.27%**	12.22%	14.44%
mas-fem-sin	462	53.1%	56.19%	54.52%	43.33%	40.95%	**95%**	85.71%	87.14%
mas-fem-plu	462	23.9%	21.69%	22.06%	6.25%	5.88%	**67.08%**	51.10%	48.53%
accuracy on syntactic	10876	42.28%	43.96%	44.11%	33.75%	34.34%	**70.13%**	67.72%	67.90%
overall accuracy	19791	55.47%	53.16%	53.49%	50.92%	50.30%	**72.09%**	39.92%	40.25%
out-of-vocabulary words		35	48	48	48	48	-	-	-

manualDICE. These vectors are visualized in Figure 2a using the t-SNE dimensionality reduction technique.

Similarly, we generated Figure 2b to illustrate how documents in RCV2 are represented by Word2Vec$_{\text{RCV}}$, GloVe$_{\text{RCV}}$, and FastText$_{\text{RCV}}$. Different colors are assigned to different classes for clarity.

As can be noticed from the figures, there is a substantial overlap among data points belonging to different categories, indicating a lack of clear separation between categories in many instances. This overlap is even more pronounced when FastText is utilized to calculate the document vectors. This observation underscores the complexity of the data and highlights potential challenges that text categorization algorithms may encounter.

C. WORD ANALOGY TEST

A preliminary test of the word embeddings was done by evaluating their ability to solve word analogies. Considering the generic analogy "A is to B as C is to D", the test searches for the word d whose embedding is the most similar to the vector obtained by the formula $B - A + C$ applied to the embeddings of these words. The Italian analogy dataset, developed by Berardi et al. [47], derives from the translation of the English Google word analogy dataset, excluding multi-word embeddings, and consists of 19,791 analogies using 1,072 words. We found two errors in the analogy dataset and modified them ("nonno" instead of "nonnno" and "kirghizistan" instead of "kirghisistan"). We tested the SOTA models and our new models trained for 50 epochs. The accuracy of each model has been calculated using the function *evaluate_word_analogies* of the Python library gensim and is reported in Table 4. The similarity function used to solve the analogies is the cosine similarity. For each line of Table 4, i.e., for each category of analogy, the higher value of accuracy is highlighted in bold. Some words in the analogies were not present in the vocabularies of the word embedding models, for FastText this is not a problem because the model is able to calculate embeddings of even out-of-vocabulary words, for Word2Vec and GloVe these words are skipped for the evaluation of the analogies.

A first observation is that FastText$_{\text{SOTA}}$ performs better than all the other models in semantic analogies and syntactic analogies. Focusing on the three Word2Vec models, it can be noticed that Word2Vec$_{\text{SOTA}}$ fails to solve all the comparative and plural-verbs (1st person) analogies while our new Word2Vec models are able to solve some of them. The performances of the three models on the other analogy categories are quite similar. The GloVe models are the ones that reported the lowest performance on syntactic analogies, failing to solve all the analogies on present and remote past verbs. While Word2Vec and GloVe exhibit a decline in performance when it comes to syntactic analogies compared to semantic ones, FastText$_{\text{SOTA}}$ maintains consistent performance in both domains. On the other hand, comparing the three FastText models, we can see that our new models recorded a very low accuracy on semantic analogies while they have comparable performances with respect to FastText$_{\text{SOTA}}$ and outperform all other models on syntactic analogies.

TABLE 5. F1-score of text categorization using TF-IDF based document vectors and unweighted word embeddings.

Algorithm	manualDICE				RCV2			
	$\text{TF-IDF}_{10,000}$	TF-IDF_{300}	$\text{Word2Vec}_{\text{SOTA}}$	$\text{FastText}_{\text{SOTA}}$	$\text{TF-IDF}_{10,000}$	TF-IDF_{300}	$\text{Word2Vec}_{\text{SOTA}}$	$\text{FastText}_{\text{SOTA}}$
Gaussian NB	-	-	68%	71%	-	-	78%	60%
Bernoulli NB	80%	**78%**	65%	71%	88%	82%	75%	73%
KNN (k=1)	74%	36%	68%	68%	91%	86%	90%	89%
KNN (k=3)	75%	48%	69%	69%	91%	87%	**91%**	89%
KNN (k=5)	77%	42%	73%	72%	92%	87%	**91%**	89%
Decision Tree	76%	70%	55%	56%	83%	81%	75%	77%
Bagging (Decision Tree)	79%	70%	67%	69%	87%	85%	85%	84%
Bagging (KNN)	77%	49%	72%	72%	92%	87%	**91%**	90%
Random Forest	80%	72%	72%	75%	90%	88%	89%	88%
Extra Trees	82%	71%	70%	71%	91%	89%	89%	88%
AdaBoost	75%	59%	-	-	91%	88%	89%	89%
XGBoost	85%	74%	80%	81%	92%	89%	**91%**	**91%**
SVC (linear)	**86%**	**78%**	82%	78%	93%	88%	90%	89%
SVC (rbf)	83%	76%	**83%**	**83%**	**94%**	**90%**	**91%**	**91%**
Perceptron	81%	69%	81%	77%	83%	82%	87%	86%

V. EXPERIMENTAL EVALUATION

We selected fifteen classifiers: Naïve Bayes with Gaussian (Gaussian NB) and multivariate Bernoulli distributions (Bernoulli NB), the K-nearest neighbors algorithm with a variable number of nearest neighbors (KNN), Decision Tree, Bagging classifier with Decision Tree (Bagging (Decision Tree)) or KNN (Bagging (KNN)) as base estimator, Random Forest, Extra Trees, Adaboost, XGBoost, SVC classifier with linear (SVC (linear)) and radial basis function (SVC (rbf)) as kernel, Perceptron linear classifier (Perceptron).

We trained them on the document vectors of the MT dataset and we tested the obtained models on the manualDICE dataset. In the same way, the classification models obtained after the training on the document vectors of the RCV2 train set have been tested on the RCV2 test set.

The experiments conducted were designed with the following objectives: (1) to assess and compare the efficacy of various machine learning classifiers, (2) to ascertain the benefits of using word embeddings with respect to TF-IDF document vectors, (3) to examine variations in text categorization performance when applying filters and/or weights to word embeddings, (4) to compare diverse training configurations for word embedding models, and (5) evaluate the impact of training word embedding models on a subset of the documents to classify.

In order to assess the efficacy of individual classifiers, we employed the weighted-average method for evaluating recall, precision, and the F1-score. This particular averaging technique takes into account the imbalance in class labels present within our datasets. It is calculated by computing the metric independently for each category and then taking their average weighted by the number of true instances for each category. Consequently, this approach may yield an F1-score that does not necessarily fall between the values of precision and recall.

A. COMPARISON OF TF-IDF AND UNWEIGHTED WORD EMBEDDINGS

This experiment aims to compare the performance of text categorization when documents are represented by TF-IDF vectors or by the document vectors of $\text{Word2Vec}_{\text{SOTA}}$ and $\text{FastText}_{\text{SOTA}}$.

The TF-IDF representation aims at assigning more importance to words that appear more frequently in a document and less in the whole dataset. We used the python class TfidfVectorizer[7] to obtain the TF-IDF representations after the tokenization and the stop word removal. The size of the vectors, i.e., the number of features to consider, can be set to any number. We decided to test the maximum dimension of 10,000 ($\text{TF-IDF}_{10,000}$) which means that the size of each document vector is at most 10,000. In addition, we performed another test reducing the number of features to 300 (TF-IDF_{300}) since the size of word embeddings of $\text{Word2Vec}_{\text{SOTA}}$ and $\text{FastText}_{\text{SOTA}}$ is 300.

Table 5 shows the evaluation on manualDICE and RCV2, the highest value for each column is reported in bold. Since the values of precision and recall are quite similar in our experiments, we report only the value of F1-score in the table. On manualDICE, the highest value of the F1-score (86%) is obtained by $\text{TF-IDF}_{10,000}$. Comparing the four methods used to obtain the document representation, it can be noticed that there is a similar trend in the F1-scores reported by the different algorithms. Indeed, XGBoost and SVC are the classifiers which reach higher values than the other algorithms in all four cases. Similar considerations can be made for the RCV2 dataset. The highest value of F1-score is 94% and is obtained when the $\text{TF-IDF}_{10,000}$ document representations are used by the SVC with the rbf kernel.

In most cases, $\text{TF-IDF}_{10,000}$ has better performances than $\text{Word2Vec}_{\text{SOTA}}$ and $\text{FastText}_{\text{SOTA}}$, especially for manualDICE. $\text{TF-IDF}_{10,000}$ is always better than TF-IDF_{300}. In the end, TF-IDF_{300} shows always lower values of F1-score except for two classifiers, the Bernoulli Naive Bayes and the Decision Tree. This behaviour suggests that reducing the vector size to 300, word embeddings are more informative than the TF-IDF based vectors.

[7]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

TABLE 6. F1-score of text categorization using weight and filter on word embeddings.

Algorithm	manualDICE						RCV2					
	Word2Vec$_{SOTA}$			FastText$_{SOTA}$			Word2Vec$_{SOTA}$			FastText$_{SOTA}$		
	OPT1	OPT2	OPT3	OPT1	OPT2	OPT3	OPT1	OPT2	OPT3	OPT1	OPT2	OPT3
Gaussian NB	58%	65%	69%	62%	67%	73%	72%	76%	77%	78%	66%	59%
Bernoulli NB	62%	64%	66%	71%	71%	72%	76%	76%	74%	76%	75%	72%
KNN (k=1)	66%	68%	69%	64%	68%	68%	89%	90%	**91%**	87%	89%	89%
KNN (k=3)	68%	71%	71%	66%	69%	72%	89%	90%	**91%**	88%	89%	89%
KNN (k=5)	71%	72%	74%	67%	71%	73%	89%	90%	**91%**	88%	89%	89%
Decision Tree	51%	52%	54%	53%	57%	59%	74%	74%	75%	74%	75%	76%
Bagging (Decision Tree)	65%	67%	67%	65%	69%	71%	84%	84%	84%	83%	84%	84%
Bagging (KNN)	70%	73%	75%	69%	71%	74%	89%	90%	**91%**	89%	89%	89%
Random Forest	72%	72%	71%	72%	75%	74%	88%	88%	89%	88%	88%	88%
Extra Trees	70%	71%	69%	68%	72%	72%	88%	88%	89%	88%	89%	88%
AdaBoost	-	-	-	-	-	-	88%	88%	89%	89%	89%	89%
XGBoost	-	-	-	-	-	82%	**91%**	**91%**	**91%**	91%	**92%**	91%
SVC (linear)	78%	81%	**82%**	83%	**84%**	79%	89%	89%	90%	90%	90%	89%
SVC (rbf)	81%	**82%**	81%	83%	**84%**	83%	90%	90%	**91%**	91%	91%	91%
Perceptron	77%	80%	81%	80%	**84%**	77%	84%	85%	88%	86%	81%	84%

B. WEIGHTING AND FILTERING WORD EMBEDDINGS

The scope of this experiment is to evaluate the impact of weighting and filtering word embeddings before averaging them to obtain the document vectors.

We defined three options to compare:

- OPT1: we weigh word embeddings using TF-IDF, in this way, more importance is given to the words that appear many times in the document and few times in the entire dataset;
- OPT2: we weigh word embeddings using TF-IDF and apply a filter to exclude words with IDF lower than the 90th percentile of all the values of IDFs of the dataset, in this way, we exclude words that appear very few times in the dataset;
- OPT3: we apply only the filter to word embeddings without weighting them.

Table 6 shows the F1-scores obtained by the classifiers on the three options when Word2Vec$_{SOTA}$ or FastText$_{SOTA}$ are used to extract the word embeddings. For each word embedding model, the highest value is reported in bold. Focusing only on Word2Vec$_{SOTA}$ and reading the table line by line, for both test datasets (`manualDICE` and `RCV2`), in most cases the highest F1-score is obtained by using OPT3. The situation is slightly different with FastText$_{SOTA}$, in particular for `RCV2`. Indeed, in this case, the performances of the three options are very similar.

Best results on `manualDICE` are reported by SVC and Perceptron, while on `RCV2` the highest F1-scores are obtained by KNN, Bagging (when KNN is used as base estimator), XGBoost and SVC.

For the other experiment, we will use OPT1.

C. WORD EMBEDDING MODEL CONFIGURATION

This experiment aims to compare the performance of the new word embedding models we trained for 20 or 50 epochs.

The training of the word embedding models has been executed on a server infrastructure made up of several nodes. Each node is equipped with a 40-core Intel Xeon Gold 6230 CPU (2.10 GHz) and 62 GB for main memory. Table 7 reports the training time for each different configuration of the three models on our two datasets. Even if the second dataset is bigger than the first one, the training time is the same on the two datasets. It is interesting to notice that GloVe is much faster than Word2Vec and FastText. This is due to the optimized approach of matrix factorization used by GloVe which is more efficient in terms of computational resources compared to the iterative methods used by Word2Vec and FastText.

In Table 8, we show the values of F1-score for each categorization algorithm. We report in bold the highest values for each dataset. The SVC algorithm with the rbf kernel reports the highest value (84%) on `manualDICE` using GloVe regardless of the number of epochs used in the training of the word embedding model, while the best F1-score on `RCV2` is 93% and is registered by XGBoost using Word2Vec trained for 20 epochs.

Comparing the F1-scores for the two configurations of each model, we do not notice improvements when word embedding models are trained for 50 epochs. There are only 6 cases of an improvement higher than 3% on the F1-score, all these cases are related to the `manualDICE` dataset, in 4 out of the 6 cases the 50-epochs trained model performs better than the 20-epochs trained model. In only one case, the F1-score increases by 18% (from 63% to 81%); this happens when the Bagging algorithm is applied using Decision Tree as a base estimator on the document representations of GloVe$_{MT}$.

D. IMPACT OF TRAINING DATASET ON WORD EMBEDDING MODELS

In this Section, we want to compare the performance of the SOTA models with respect to the models we trained on a

TABLE 7. Training time (in hours) of the word embedding models.

Training dataset	Word2Vec		GloVe		FastText	
	20 epochs	50 epochs	20 epochs	50 epochs	20 epochs	50 epochs
Italian Wikipedia, webz and MT	21	55	< 1	2	13	34
Italian Wikipedia, webz and RCV2 train	21	55	< 1	2	13	34

TABLE 8. F1-score of text categorization using word embedding models trained for 20 or 50 epochs.

Algorithm	`manualDICE`						`RCV2`					
	Word2Vec$_{MT}$		GloVe$_{MT}$		FastText$_{MT}$		Word2Vec$_{RCV}$		GloVe$_{RCV}$		FastText$_{RCV}$	
	20 ep.	50 ep.	20 ep.	50 ep.	20 ep.	50 ep.	20 ep.	50 ep.	20 ep.	50 ep.	20 ep.	50 ep.
Gaussian NB	65%	66%	67%	69%	48%	51%	80%	80%	77%	76%	72%	71%
Bernoulli NB	67%	67%	68%	69%	51%	54%	79%	80%	77%	77%	70%	68%
KNN (k=1)	71%	72%	66%	67%	58%	57%	92%	91%	90%	91%	87%	87%
KNN (k=3)	72%	73%	69%	69%	58%	59%	92%	92%	91%	91%	87%	88%
KNN (k=5)	76%	76%	71%	72%	58%	63%	92%	92%	91%	91%	88%	88%
Decision Tree	57%	63%	51%	52%	50%	48%	83%	82%	79%	78%	73%	74%
Bagging (Decision Tree)	69%	71%	63%	81%	64%	62%	88%	88%	85%	85%	82%	83%
Bagging (KNN)	76%	77%	72%	80%	62%	63%	92%	92%	91%	91%	88%	88%
Random Forest	77%	76%	69%	68%	67%	66%	90%	91%	89%	89%	86%	87%
Extra Trees	75%	74%	66%	66%	65%	63%	91%	91%	89%	89%	86%	87%
AdaBoost	60%	62%	59%	54%	52%	49%	-	85%	81%	-	86%	83%
XGBoost	81%	81%	78%	72%	76%	75%	**93%**	92%	92%	92%	90%	90%
SVC (linear)	83%	83%	82%	81%	77%	80%	91%	91%	90%	91%	89%	90%
SVC (rbf)	83%	83%	**84%**	**84%**	78%	79%	91%	91%	92%	92%	89%	91%
Perceptron	82%	83%	82%	80%	81%	81%	89%	89%	88%	88%	88%	89%

TABLE 9. F1-score of the best classifiers on manualDICE.

Algorithm	Word2Vec		FastText	
	SOTA	50 epochs	SOTA	50 epochs
SVC (linear)	82%	**83%**	78%	80%
SVC (rbf)	**83%**	**83%**	**83%**	79%
Perceptron	81%	**83%**	77%	81%

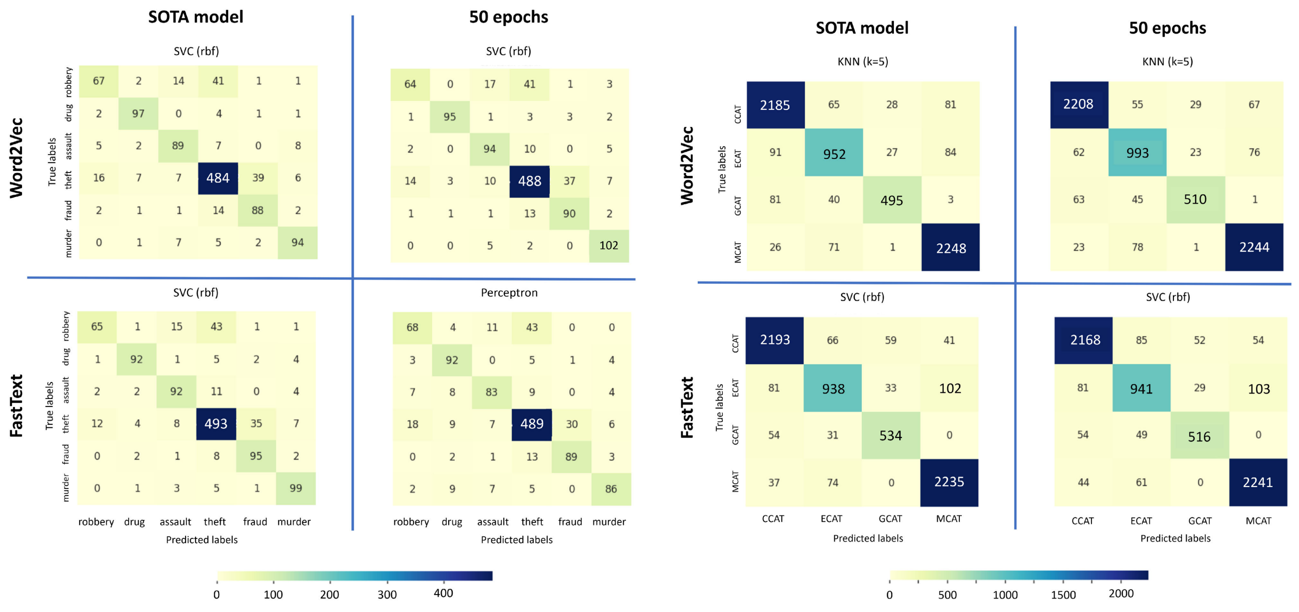
TABLE 10. F1-score of the best classifiers on RCV2.

Algorithm	Word2Vec		FastText	
	SOTA	50 epochs	SOTA	50 epochs
KNN (k=1)	90%	91%	89%	87%
KNN (k=3)	91%	**92%**	89%	88%
KNN (k=5)	91%	**92%**	89%	88%
Bagging (KNN)	91%	**92%**	90%	88%
SVC (linear)	90%	91%	89%	90%
SVC (rbf)	91%	91%	91%	91%

dataset containing documents very similar to the documents to classify. For this reason, in Table 9, we report the best results of F1-score for `manualDICE` categorization obtained by Word2Vec$_{SOTA}$ and FastText$_{SOTA}$, and Word2Vec$_{MT}$ and FastText$_{MT}$ trained for 50 epochs. In the same way, Table 10 shows the best results on the `RCV2` dataset. The highest values for each dataset are highlighted in bold. In both cases, SVC is among the best algorithms. Focusing on Word2Vec and considering one algorithm at a time, we can notice that the performances of the two models are very similar with slightly higher F1-score values when the ad-hoc trained model is used. While with FastText, the performance is slightly better when the SOTA model is used.

In order to understand the behaviour of the categorization, we tried to analyze in detail the results on each category of the test datasets. For both datasets, we selected the best algorithm or one of the best for each word embedding model. Figure 3 shows the confusion matrices obtained in each of these cases. In all four cases relating to the `manualDICE` dataset, it can be noticed that the highest number of errors occur in two cases: prediction of "theft" while the true label is "robbery", and prediction of "fraud" when the true label is "theft". This behaviour could be due to the unbalance of the dataset since the "theft" class is the predominant class in `manualDICE` and also in `MT`, while "robbery" and "fraud" are the two classes less frequent in `manualDICE`. It is interesting also to notice that the class with the lowest number of false positives and false negatives is "drug" in all four cases, this class is also the second most frequent class in the `MT` dataset and the percentage of its frequency in the test dataset is very low (just 6%) with respect to the whole dataset composed of `manualDICE` and `MT`.

For the `RCV2` dataset, the ratio between the documents in the training set and in the test set for each category is always the same. In this way, there is no category that is very frequent in the training set and very infrequent in the test set, or vice versa. Comparing the different word embedding models, we observe that the sum of false positives and false negatives for each category is almost the same. The categories with the highest number of errors are "CCAT" and "ECAT",



(a) manualDICE (b) RCV2

FIGURE 3. Confusion matrix of the best classifiers.

however, considering the percentage of errors with respect to the total number of instances of each class in the RCV2 dataset, we discovered that the categories with the highest error percentage (around 30%) are "ECAT" and "GCAT" which are the less frequent categories in the dataset.

VI. DISCUSSION

In this study, our primary goal was to compare various methodologies for classifying Italian news articles into predefined categories. To achieve this, we implemented multiple pipelines to acquire document representations, trained fifteen classifiers, and assessed their performance based on precision, recall, and F1-score. Our evaluation encompassed the use of TF-IDF document vectors and word embeddings generated by Word2Vec, GloVe, and FastText.

Specifically, for each model, we conducted two training sessions: one for 20 epochs and another for 50 epochs,, both initiated from scratch. Additionally, for Word2Vec and FastText, we examined the performance of the state-of-the-art Italian models. Two distinct datasets were employed for testing, one comprising six predefined categories and the other with four.

In total, we conducted approximately 500 experiments to thoroughly investigate and compare the effectiveness of the various approaches employed in this study.

The results of our experiments reveal several noteworthy findings:

- **TF-IDF vs. Word Embeddings**: text categorization based on TF-IDF vectors can demonstrate a slight performance advantage over the use of word embeddings for document representations when the dimensionality of the TF-IDF vectors (10,000 features) significantly exceeds that of the word embeddings (300 features). The extent of this performance difference can vary considerably depending on the specific dataset under consideration. In our experiments, the highest F1-score achieved by TF-IDF with 10,000 features on manualDICE is 86%, whereas the highest F1-score obtained using word embeddings is 84%. Similarly, for RCV2, these respective values are 94% and 93%.
- **Machine Learning Classifiers**: Comparing fifteen machine learning classifiers, regardless of the approach employed to find the document vectors, we can identify a similar trend in the categorization performance. The SVC algorithm has very good performance in text categorization since it achieves high F1 scores in both datasets we used in our experiments, while Gaussian Naive Bayes, Bernoulli Naive Bayes, and Decision Tree report low F1 scores. Therefore, we could consider these algorithms not suitable for text categorization.
- **Precision and Recall**: In all our experiments, precision and recall have almost the same value. This means that for each category, the number of false positives is similar to the number of false negatives. In other words, for each class, the categorization models have the same ability to identify its correct instances and avoid classifying other instances in that class, erroneously.
- **Dataset Size Impact**: Regardless of the method used to obtain the document representation, we noticed higher F1-scores on the RCV2 dataset with respect to the manualDICE dataset. This may be due to the size of the two datasets since RCV2 is three times bigger

than `MT+manualDICE`, thus the classifiers used for `RCV2` are trained on more documents with respect to the models used for `manualDICE`.

- **Categorization errors** (i.e., the sum of false positives and false negatives for each category) occur most in the less represented categories or in the categories with a strong imbalance in the number of instances in the test set with respect to the training set. We tested an approach of data augmentation to compensate for the low number of instances in certain categories. In particular, we used the Synthetic Minority Oversampling TEchnique (SMOTE) [80] to create synthetic document vectors for the less represented categories. However, this approach did not improve the categorization results as we expected. For this reason, we did not discuss in detail the results obtained.
- **Training epochs impact**: Comparing the results obtained using the new word embedding models we trained for 20 or 50 epochs, we did not observe the performance improvement we expected when the number of epochs increased. Maybe, this is due to the value of the learning rate (0.03) we used when training the word embedding models. A possible solution to improve the performance of these models could be to increase the learning rate.
- **New FastText models' poor performance**: FastText$_{\text{MT}}$ and FastText$_{\text{RCV}}$, regardless of the number of epochs used, underperform compared to FastText$_{\text{SOTA}}$ and all the Word2Vec and GloVe models. This result is confirmed by the poor performance of these models also in the word analogy test and the strong overlap of FastText$_{\text{MT}}$ and FastText$_{\text{RCV}}$ word embeddings shown in Figure 2. The reason for this behavior could be related to the choice of hyperparameters, including the learning rate, the n-gram dimension, and the number of epochs. An alternative approach could be the use of the automatic hyperparameter optimization implemented by Facebook researchers in the Python library fasttext.[8] It is noteworthy to notice also that the datasets used to train FastText$_{\text{MT}}$ and FastText$_{\text{RCV}}$ are much smaller than the dataset used for FastText$_{\text{SOTA}}$. We are unable to find a clear explanation for the superior performance of Word2Vec and GloVe when they are trained from scratch on the same dataset as FastText.

VII. CONCLUSION

This study has provided valuable insights into the classification of Italian news articles into predefined categories. The research underscores the importance of well-constructed word embedding models and robust machine learning algorithms in the context of document categorization for the Italian language.

Through extensive experimentation, we have explored various document representation methods and machine learning classifiers, shedding light on their respective strengths and weaknesses. Our contributions extend beyond the research domain, facilitating the broader community's access to high-quality Italian word embedding models (since six new models have been released). Additionally, the insights gained from our comparative evaluation of machine learning algorithms provide practical guidance for researchers and practitioners seeking to tackle similar text classification challenges.

There can be a performance advantage of using TF-IDF-based vectors w.r.t. word embeddings, and this becomes apparent when working with very higher-dimensional vectors. Usually, the Support Vector Classifier (SVC) is a robust performer in text categorization, while Gaussian Naive Bayes, Bernoulli Naive Bayes, and Decision Tree should be used with caution in this context. However, the classifiers are negatively affected by class imbalances in train and test datasets. Also, the dataset size has an impact on the categorization performance, with larger datasets yielding more favorable results. Our experiments indicate that training word embedding models for a high number of epochs (i.e., 50 epochs) does not confer any advantage in text categorization. There are still some open issues, like the hyperparameters optimization when training word embedding models, that could improve the performance of text categorization, especially if a high number of epochs is used.

Looking ahead, our study paves the way for future investigations into more advanced techniques, such as Transformer-based models, from autoencoding LLMs like BERT [21] or ELECTRA [81] to autoregressive LLMs like GPT models [82], open-source Llama 2 models [83], [84] from Meta or the models from Mistral AI (e.g., Mistral 7B [85]) which hold promise in further enhancing the accuracy and sophistication of Italian text classification tasks [86], [87], [88], [89], [90], [91]. Other future directions of our research will include the use of multi-label categorization models in news articles, where articles often straddle multiple categories.

We believe this study lays the foundation for continued exploration and improvement in the field of news article classification and opens the street for an in-depth evaluation of Italian models that are still not very common but are necessary for dealing with Italian corpora. Moreover, the released models can benefit other NLP tasks beyond news categorization, such as sentiment analysis, entity recognition, or machine translation.

[8]FastText automatic hyperparameter optimization: https://fasttext.cc/docs/en/autotune.html

REFERENCES

[1] P. Henshall and D. Ingram, *The News Manual: A Training Book for Journalists*. Sydney, NSW, Australia: Poroman Press, 1991.

[2] H. Pöttker, "News and its communicative quality: The inverted pyramid—When and why did it appear?" *Journalism Stud.*, vol. 4, no. 4, pp. 501–511, Nov. 2003, doi: 10.1080/1461670032000136596.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. 1st Int. Conf. Learn. Represent.*, Scottsdale, AZ, USA, Y. Bengio and Y. LeCun, Eds., 2013.

[4] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, A. Moschitti, B. Pang, and W. Daelemans, Eds., 2014, pp. 1532–1543, doi: 10.3115/v1/d14-1162.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017, doi: 10.1162/tacl_a_00051.

[6] F. Rollo, G. Bonisoli, and L. Po, "Italian Word2Vec models," Univ. Modena Reggio Emilia, Modena, Italy, Oct. 2023, doi: 10.5281/zenodo.8367726.

[7] F. Rollo, G. Bonisoli, and L. Po, "Italian FastText models," Univ. Modena Reggio Emilia, Modena, Italy, Oct. 2023, doi: 10.5281/zenodo.8426107.

[8] F. Rollo, G. Bonisoli, and L. Po, "Italian GloVe models," Univ. Modena Reggio Emilia, Modena, Italy, Oct. 2023, doi: 10.5281/zenodo.8427306.

[9] F. Rollo, G. Bonisoli, and L. Po, "Supervised and unsupervised categorization of an imbalanced Italian crime news dataset," in *Proc. Conf. Inf. Syst. Manag.*, in Lecture Notes in Business Information Processing, vol. 442, E. Ziemba and W. Chmielarz, Eds. Cham, Switzerland: Springer, 2021, pp. 117–139, doi: 10.1007/978-3-030-98997-2.

[10] G. Bonisoli, F. Rollo, and L. Po, "Using word embeddings for Italian crime news categorization," in *Proc. 16th Conf. Comput. Sci. Intell. Syst. (FedCSIS)*, in Annals of Computer Science and Information Systems, M. Ganzha, L. A. Maciaszek, M. Paprzycki, and D. Slezak, Eds., Sep. 2021, pp. 461–470, doi: 10.15439/2021F118.

[11] D. Rogers, A. Preece, M. Innes, and I. Spasic, "Real-time text classification of user-generated content on social media: Systematic review," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 4, pp. 1154–1166, Aug. 2022, doi: 10.1109/TCSS.2021.3120138.

[12] A. Dhar, H. Mukherjee, N. S. Dash, and K. Roy, "Text categorization: Past and present," *Artif. Intell. Rev.*, vol. 54, no. 4, pp. 3007–3054, Apr. 2021.

[13] V. K. Vijayan, K. R. Bindu, and L. Parameswaran, "A comprehensive study of text classification algorithms," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Udupi, India, Sep. 2017, pp. 1109–1113, doi: 10.1109/ICACCI.2017.8125990.

[14] X. Chen, P. Cong, and S. Lv, "A long-text classification method of Chinese news based on BERT and CNN," *IEEE Access*, vol. 10, pp. 34046–34057, 2022, doi: 10.1109/ACCESS.2022.3162614.

[15] K. M. Hasib, S. Azam, A. Karim, A. A. Marouf, F. M. J. M. Shamrat, S. Montaha, K. C. Yeo, M. Jonkman, R. Alhajj, and J. G. Rokne, "MCNN-LSTM: Combining CNN and LSTM to classify multi-class text in imbalanced news data," *IEEE Access*, vol. 11, pp. 93048–93063, 2023, doi: 10.1109/ACCESS.2023.3309697.

[16] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, in Association for Computational Linguistics, Vancouver, BC, Canada, R. Barzilay and M. Kan, Eds., 2017, pp. 562–570.

[17] A. B. Dieng, C. Wang, J. Gao, and J. W. Paisley, "TopicRNN: A recurrent neural network with long-range semantic dependency," in *Proc. 5th Int. Conf. Learn. Represent.*, Toulon, France, 2017. [Online]. Available: https://openreview.net/forum?id=rJbbOLcex

[18] M. T. Vidal, E. S. Rodríguez, and J. A. Reyes-Ortíz, "Classification of criminal news over time using bidirectional LSTM," in *Proc. Int. Conf. Pattern Recognit. Artif. Intell.*, in Lecture Notes in Computer Science, vol. 12068, Zhongshan, China, Y. Lu, N. Vincent, P. C. Yuen, W. Zheng, F. Cheriet, and C. Y. Suen, Eds. Cham, Switzerland: Springer, 2020, pp. 702–713.

[19] N. Shi, Z. Chen, L. Chen, and R. S. T. Lee, "CNO-LSTM: A chaotic neural oscillatory long short-term memory model for text classification," *IEEE Access*, vol. 10, pp. 129564–129579, 2022, doi: 10.1109/ACCESS.2022.3228600.

[20] Y. Zhang, "Research on text classification method based on LSTM neural network model," in *Proc. IEEE Asia–Pacific Conf. Image Process., Electron. Comput. (IPEC)*, Apr. 2021, pp. 1019–1022.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, in Association for Computational Linguistics, vol. 1. Minneapolis, MN, USA, J. Burstein, C. Doran, and T. Solorio, Eds., Jun. 2019, pp. 4171–4186.

[22] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 5754–5764. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e6 6733e9ee67cc69-Abstract.html

[23] X. She, J. Chen, and G. Chen, "Joint learning with BERT-GCN and multi-attention for event text classification and event assignment," *IEEE Access*, vol. 10, pp. 27031–27040, 2022, doi: 10.1109/ACCESS.2022.3156918.

[24] A. Sanwaliya, K. Shanker, and S. C. Misra, "Categorization of news articles: A model based on discriminative term extraction method," in *Proc. 2nd Int. Conf. Adv. Databases, Knowl., Data Appl.*, Menuires, France, F. Laux and L. Strömbäck, Eds., Apr. 2010, pp. 149–154.

[25] M. M. Al-Tahrawi, "Arabic text categorization using logistic regression," *Int. J. Intell. Syst. Appl.*, vol. 7, no. 6, pp. 71–78, May 2015.

[26] R. Wongso, F. A. Luwinda, B. C. Trisnajaya, and O. Rusli, "News article text classification in Indonesian language," in *Proc. ICCSCI*, 2017, pp. 137–143.

[27] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features," in *Proc. IEEE 14th Int. Conf. Cognit. Informat. Cognit. Comput.*, Beijing, China, N. Ge, J. Lu, Y. Wang, N. Howard, P. Chen, X. Tao, B. Zhang, and L. A. Zadeh, Eds., Jul. 2015, pp. 136–140, doi: 10.1109/ICCI-CC.2015.7259377.

[28] C. Pan, J. Huang, J. Gong, and X. Yuan, "Few-shot transfer learning for text classification with lightweight word embedding based models," *IEEE Access*, vol. 7, pp. 53296–53304, 2019.

[29] W. Wang, G. He, and X. Liu, "Text multi-classification based on word embedding and multi-grained cascade forest," in *Proc. IEEE 5th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2019, pp. 13–17.

[30] C. Wang, P. Nulty, and D. Lillis, "A comparative study on word embeddings in deep learning for text classification," in *Proc. 4th Int. Conf. Natural Lang. Process. Inf. Retr.*. New York, NY, USA, Dec. 2020, pp. 37–46.

[31] M. Víta and V. Kríž, "Word2vec based system for recognizing partial textual entailment," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Poland, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., Sep. 2016, pp. 513–516.

[32] A. Moreo, A. Esuli, and F. Sebastiani, "Word-class embeddings for multiclass text classification," *Data Mining Knowl. Discovery*, vol. 35, no. 3, pp. 911–963, May 2021, doi: 10.1007/s10618-020-00735-3.

[33] A. Fesseha, S. Xiong, E. D. Emiru, M. Diallo, and A. Dahou, "Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya," *Information*, vol. 12, no. 2, p. 52, Jan. 2021, doi: 10.3390/info12020052.

[34] A. Borg, M. Boldt, O. Rosander, and J. Ahlstrand, "E-mail classification with machine learning and word embeddings for improved customer support," *Neural Comput. Appl.*, vol. 33, no. 6, pp. 1881–1902, Mar. 2021, doi: 10.1007/s00521-020-05058-4.

[35] E. Christodoulou, A. Gregoriades, M. Pampaka, and H. Herodotou, "Application of classification and word embedding techniques to evaluate tourists' hotel-revisit intention," in *Proc. 23rd Int. Conf. Enterprise Inf. Syst.*, J. Filipe, M. Smialek, A. Brodsky, and S. Hammoudi, Eds., 2021, pp. 216–223, doi: 10.5220/0010453502160223.

[36] P. Semberecki and H. Maciejewski, "Deep learning methods for subject text classification of articles," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Prague, Czech Republic, Sep. 2017, pp. 357–360.

[37] D. Chandrasekaran and V. Mago, "Comparative analysis of word embeddings in assessing semantic similarity of complex sentences," *IEEE Access*, vol. 9, pp. 166395–166408, 2021, doi: 10.1109/ACCESS.2021.3135807.

[38] Y. Didi, A. Walha, and A. Wali, "COVID-19 tweets classification based on a hybrid word embedding method," *Big Data Cognit. Comput.*, vol. 6, no. 2, p. 58, May 2022, doi: 10.3390/bdcc6020058.

[39] H. A. Almuzaini and A. M. Azmi, "Impact of stemming and word embedding on deep learning-based Arabic text categorization," *IEEE Access*, vol. 8, pp. 127913–127928, 2020, doi: 10.1109/ACCESS.2020.3009217.

[40] T. Nugent, F. Petroni, N. Raman, L. Carstens, and J. L. Leidner, "A comparison of classification models for natural disaster and critical event detection from news," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Boston, MA, USA, Dec. 2017, pp. 3750–3759, doi: 10.1109/BIGDATA.2017.8258374.

[41] I. Kaibi, E. H. Nfaoui, and H. Satori, "A comparative evaluation of word embeddings techniques for Twitter sentiment analysis," in *Proc. Int. Conf. Wireless Technol., Embedded Intell. Syst. (WITS)*, Apr. 2019, pp. 1–4.

[42] I. Guellil, A. Adeel, F. Azouaou, S. Chennoufi, H. Maafi, and T. Hamitouche, "Detecting hate speech against politicians in Arabic community on social media," *Int. J. Web Inf. Syst.*, vol. 16, no. 3, pp. 295–313, Jul. 2020, doi: 10.1108/ijwis-08-2019-0036.

[43] L. Sravani, A. S. Reddy, and S. Thara, "A comparison study of word embedding for detecting named entities of code-mixed data in Indian language," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2018, pp. 2375–2381.

[44] N. Hartmann, E. Fonseca, C. Shulby, M. Treviso, J. Silva, and S. Aluísio, "Portuguese word embeddings: Evaluating on word analogies and natural language tasks," in *Proc. 11th Brazilian Symp. Inf. Hum. Lang. Technol.*, Oct. 2017, pp. 122–131. [Online]. Available: https://www.aclweb.org/anthology/W17-6615

[45] W. Ling, C. Dyer, A. W. Black, and I. Trancoso, "Two/too simple adaptations of Word2Vec for syntax problems," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, Denver, CO, USA, 2015, pp. 1299–1304. [Online]. Available: https://aclanthology.org/N15-1142

[46] R. Tripodi and S. L. Pira, "Analysis of Italian word embeddings," in *Proc. 4th Italian Conf. Comput. Linguistics*, Rome, Italy, R. Basili, M. Nissim, and G. Satta, Eds., 2017. [Online]. Available: https://ceur-ws.org/Vol-2006/paper045.pdf

[47] G. Berardi, A. Esuli, and D. Marcheggiani, "Word embeddings go to Italy: A comparison of models and training datasets," in *Proc. 6th Italian Inf. Retr. Workshop*, vol. 1404, Cagliari, Italy, P. Boldi, R. Perego, and F. Sebastiani, Eds., 2015. [Online]. Available: https://ceur-ws.org/Vol-1404/paper_11.pdf

[48] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," in *Proc. 1st Workshop Evaluating Vector-Space Represent. NLP*, 2016, pp. 30–35. [Online]. Available: https://aclanthology.org/W16-2506

[49] A. dos S. R. da Silva and N. T. Roman, "No place for hate speech @ AMI: Convolutional neural network and word embedding for the identification of misogyny in Italian (short paper)," in *Proc. 7th Eval. Campaign Natural Lang. Process. Speech Tools Italian. Final Workshop*, vol. 2765, V. Basile, D. Croce, M. D. Maro, and L. C. Passaro, Eds., 2020. [Online]. Available: https://ceur-ws.org/Vol-2765/paper166.pdf

[50] G. Gambino and R. Pirrone, "Investigating embeddings for sentiment analysis in Italian," in *Proc. 3rd Workshop Natural Lang. Artif. Intell. Located 18th Int. Conf. Italian Assoc. Artif. Intell.*, vol. 2521, Rende, Italy, M. Alam, V. Basile, F. Dell'Orletta, M. Nissim, and N. Novielli, Eds., 2019. [Online]. Available: https://ceur-ws.org/Vol-2521/paper-03.pdf

[51] H. Zhang and D. Li, "Naïve Bayes text classifier," in *Proc. IEEE Int. Conf. Granular Comput.*, Apr. 2007, p. 708.

[52] I. Dawar and N. Kumar, "Text categorization by content using Naïve Bayes approach," in *Proc. 11th Int. Conf. Internet Everything, Microw. Eng., Commun. Netw. (IEMECON)*, Feb. 2023, pp. 1–6.

[53] M. Granik and V. Mesyura, "Fake news detection using Naïve Bayes classifier," in *Proc. IEEE 1st Ukraine Conf. Electr. Comput. Eng. (UKRCON)*, May 2017, pp. 900–903.

[54] A. McCallum and K. Nigam, "A comparison of event models for Naïve Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, 1998, vol. 752, no. 1, pp. 41–48. [Online]. Available: http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf

[55] W. Zhang and F. Gao, "An improvement to Naïve Bayes for text classification," *Proc. Eng.*, vol. 15, pp. 2160–2164, Jan. 2011.

[56] P. Soucy and G. Mineau, "A simple KNN algorithm for text categorization," in *Proc. IEEE Int. Conf. Data Mining*, Oct. 2001, pp. 647–648.

[57] X.-P. Yu and X.-G. Yu, "Novel text classification based on K-nearest neighbor," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2007, pp. 3425–3430.

[58] H. Li, H. Jiang, D. Wang, and B. Han, "An improved KNN algorithm for text classification," in *Proc. 8th Int. Conf. Instrum. Meas., Comput., Commun. Control (IMCCC)*, Jul. 2018, pp. 1081–1085.

[59] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Trans. Geosci. Electron.*, vol. GE-15, no. 3, pp. 142–147, Jul. 1977.

[60] F. Harrag, E. El-Qawasmeh, and P. Pichappan, "Improving Arabic text categorization using decision trees," in *Proc. 1st Int. Conf. Networked Digit. Technol.*, Jul. 2009, pp. 110–115.

[61] K. Khan, R. U. Khan, A. Alkhalifah, and N. Ahmad, "Urdu text classification using decision trees," in *Proc. 12th Int. Conf. High-Capacity Opt. Netw. Enabling/Emerging Technol. (HONET)*, Dec. 2015, pp. 1–4.

[62] B. Zhang, "News text classification algorithm based on machine learning technology," in *Proc. Int. Conf. Educ., Netw. Inf. Technol. (ICENIT)*, Sep. 2022, pp. 182–186.

[63] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification Regression Trees*. New York, NY, USA: Taylor & Francis, 1984. [Online]. Available: https://books.google.es/books?id=JwQx-WOmSyQC

[64] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/bf00058655.

[65] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/a:1010933404324.

[66] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006, doi: 10.1007/s10994-006-6226-1.

[67] R. E. Schapire, "A brief introduction to boosting," in *Proc. 16th Int. Joint Conf. Artif. Intell.*, vol. 2. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1401–1406.

[68] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems*, vol. 12, S. Solla, T. Leen, and K. Müller, Eds. Cambridge, MA, USA: MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/file/96a93ba89a5b5c6c226e49b88973f46e-Paper.pdf

[69] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[70] Z. Liu, X. Lv, K. Liu, and S. Shi, "Study on SVM compared with the other text classification methods," in *Proc. 2nd Int. Workshop Educ. Technol. Comput. Sci.*, vol. 1, Mar. 2010, pp. 219–222.

[71] V. Manoj, T. Devi, and M. Kalaiyarasi, "Classification of newspaper article classification by employing support vector machine in comparison with perceptron to improve accuracy," in *Proc. 8th Int. Conf. Sci. Technol. Eng. Math. (ICONSTEM)*, Apr. 2023, pp. 1–6.

[72] M. Fanjin, H. Ling, T. Jing, and W. Xinzheng, "The research of semantic kernel in SVM for Chinese text classification," in *Proc. 2nd Int. Conf. Intell. Inf. Process.* New York, NY, USA: Association for Computing Machinery, Jul. 2017, pp. 1–7, doi: 10.1145/3144789.3144801.

[73] M. Goudjil, M. Koudil, N. Hammami, M. Bedda, and M. Alruily, "Arabic text categorization using SVM active learning technique: An overview," in *Proc. World Congr. Comput. Inf. Technol. (WCCIT)*, Jun. 2013, pp. 1–2.

[74] L. Lei and G. Qiao, "Text categorization using SVM with exponent weighted ACO," in *Proc. 31st Chin. Control Conf.*, Jul. 2012, pp. 3763–3768.

[75] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.

[76] G. Bonisoli, M. P. Di Buono, L. Po, and F. Rollo, "DICE: A dataset of Italian crime event news," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Taipei, Taiwan, H. Chen, W. E. Duh, H. Huang, M. P. Kato, J. Mothe, and B. Poblete, Eds., Jul. 2023, pp. 2985–2995, doi: 10.1145/3539618.3591904.

[77] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, Dec. 2004.

[78] G. Di Gennaro, A. Buonanno, A. Di Girolamo, A. Ospedale, F. A. N. Palmieri, and G. Fedele, "An analysis of word2vec for the Italian language," in *Progresses in Artificial Intelligence and Neural Systems*, A. Esposito, M. Faundez-Zanuy, F. C. Morabito, and E. Pasero, Eds. Singapore: Springer, 2021, pp. 137–146.

[79] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proc. 11th Int. Conf. Lang. Resour. Eval.*, Miyazaki, Japan, May 2018, pp. 3483–3487. [Online]. Available: https://aclanthology.org/L18-1550

[80] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[81] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," in *Proc. ICLR*, 2020, pp. 1–14. [Online]. Available: https://openreview.net/pdf?id=r1xMH1BtvB

[82] T. Brown et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 1877–1901.
[83] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.
[84] P. Basile, E. Musacchio, M. Polignano, L. Siciliani, G. Fiameni, and G. Semeraro, "LLaMAntino: LLaMA 2 models for effective text generation in Italian language," 2023, *arXiv:2312.09993*.
[85] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. Le Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7B," 2023, *arXiv:2310.06825*.
[86] S. Zhang, H. Yu, and G. Zhu, "An emotional classification method of Chinese short comment text based on ELECTRA," *Connection Sci.*, vol. 34, no. 1, pp. 254–273, Dec. 2022, doi: 10.1080/09540091.2021.1985968.
[87] R. Silva Barbon and A. T. Akabane, "Towards transfer learning techniques—BERT, DistilBERT, BERTimbau, and DistilBERTimbau for automatic text classification from different languages: A case study," *Sensors*, vol. 22, no. 21, p. 8184, Oct. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/21/8184
[88] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang, "Text classification via large language models," in *Proc. Findings Assoc. Comput. Linguistics: EMNLP*. Singapore: Association for Computational Linguistics, 2023, pp. 8990–9005.
[89] Y.-S. Chuang, Y. Wu, D. Gupta, R. Uppaal, A. Kumar, L. Sun, M. N. Sreedhar, S. Yang, T. T. Rogers, and J. Hu, "Evolving domain adaptation of pretrained language models for text classification," 2023, *arXiv:2311.09661*.
[90] A. Pal, S. Rajanala, R. C.-W. Phan, and K. Wong, "Self supervised BERT for legal text classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.
[91] K. Zhang, X. Hei, R. Fei, Y. Guo, and R. Jiao, "Cross-domain text classification based on BERT model," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, C. S. Jensen, E.-P. Lim, D.-N. Yang, C.-H. Chang, J. Xu, W.-C. Peng, J.-W. Huang, and C.-Y. Shen, Eds. Cham, Switzerland: Springer, 2021, pp. 197–208.

FEDERICA ROLLO received the M.S. degree in computer science engineering from the University of Modena and Reggio Emilia, Italy, in 2018, and the Ph.D. degree from the International Doctorate School in Information and Communication Technologies, University of Modena and Reggio Emilia, in 2022.

She is currently a non-tenured Assistant Professor with the Department of Engineering "Enzo Ferrari," University of Modena and Reggio Emilia. She is also a member of the Big Data Research Group (https://dbgroup.unimore.it/). During her academic career, she collaborated with several international researchers. From 2018 to 2021, she was the Task Leader of the European Project "TRAFAIR-Understanding Traffic Flows to Improve Air Quality," co-financed by INEA CEF Telecom, where she implemented a smart city data platform and managed big data streams of IoT technologies. Since February 2023, she has been working on the "ECOSISTER" project, where she collaborates to model urban mobility and implement algorithms of multimodal routing. Her current research interests include natural language processing, event extraction from news articles, and large language models.

GIOVANNI BONISOLI received the bachelor's and master's degrees in computer engineering from the University of Modena and Reggio Emilia, Modena, Italy, where he is currently pursuing the Ph.D. degree with the International Doctorate School in Information and Communication Technologies (ICT). He is focused on the use of natural language processing techniques to extract information about valuable events from web data streams. His research interests include natural language processing and event extraction.

LAURA PO is currently a member of the Board of the International Doctoral School in Information and Communication Technologies (ICT), University of Modena and Reggio Emilia, and a member of two Interdepartmental Research Centers, UNIMORE: DHMore on Digital Humanities and AIRI, the Artificial Intelligence Research and Innovation Center. Since 2020, she has been an Associate Professor with the Department of Engineering "Enzo Ferrari," University of Modena and Reggio Emilia. She is also a Senior Researcher with the Big Data Research Group (https://dbgroup.unimore.it/) and a Distinguished Web and Data Scientist in Modena, Italy. She is the author of more than 60 publications in scientific journals and international conference proceedings. Since 2009, she has been the Co-Founder of DataRiver srl, which provides data integration solutions using semantic web techniques. Throughout her career, she has led the European Project "TRAFAIR-Understanding Traffic Flows to Improve Air Quality," co-financed by INEA CEF Telecom. She has been actively involved in numerous national and European research initiatives. She is involved in two significant national projects, such as "ECOSISTER," where she works on urban mobility modeling for a carbon-neutral society, and "Lively Ageing," which aims to create a digital twin for the elderly to improve their quality of life. Her research interests include information representation and management, with a focus on spanning big dataset visualization, web and information systems, AI techniques, and knowledge and ontology design.

Open Access funding provided by 'Università degli Studi di Modena e Reggio Emilia' within the CRUI CARE Agreement