





# Creating Classification Models from Textual Descriptions of Companies Using Crunchbase

Marco Felgueiras<sup>1</sup>, Fernando Batista<sup>1,2</sup>(✉) , and Joao Paulo Carvalho<sup>2,3</sup> 

<sup>1</sup> ISCTE - Instituto Universitário de Lisboa, Lisbon, Portugal  
mfms@iscte-iul.pt

<sup>2</sup> INESC-ID Lisboa, Lisbon, Portugal  
{fmm,joao.carvalho}@inesc-id.pt

<sup>3</sup> Universidade de Lisboa, Lisbon, Portugal

**Abstract.** This paper compares different models for multilabel text classification, using information collected from Crunchbase, a large database that holds information about more than 600000 companies. Each company is labeled with one or more categories, from a subset of 46 possible categories, and the proposed models predict the categories based solely on the company textual description. A number of natural language processing strategies have been tested for feature extraction, including stemming, lemmatization, and part-of-speech tags. This is a highly unbalanced dataset, where the frequency of each category ranges from 0.7% to 28%. Our findings reveal that the description text of each company contain features that allow to predict its area of activity, expressed by its corresponding categories, with about 70% precision, and 42% recall. In a second set of experiments, a multiclass problem that attempts to find the most probable category, we obtained about 67% accuracy using SVM and Fuzzy Fingerprints. The resulting models may constitute an important asset for automatic classification of texts, not only consisting of company descriptions, but also other texts, such as web pages, text blogs, news pages, etc.

**Keywords:** Text mining · Multilabel classification · Text classification · Document classification · Machine learning · Crunchbase

## 1 Introduction

We live in a digital society where data grows day by day, most of it consisting of unstructured textual data. This creates the need of processing all this data in order to be able to collect useful information from it. Text classification may be considered a relatively simple task, but it plays a fundamental role in a variety of systems that process textual data. E-mail spam detection is one of

---

This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020.

the most well-known applications of text classification, where the main goal consists of automatically assigning one of two possible labels (spam or ham) to each message. Other well-known text classification tasks, nowadays receiving increasingly importance, include sentiment analysis and emotion detection, that consist of assign a positive/negative sentiment or an emotion to a text (e.g. happiness, anger, sadness, ...).

Crunchbase is the largest companies' database in the world, containing a large variety of up-to-date information about each company. Founded in 2007 by Michael Arrington, originally, it was the data storage for its mother company TechCrunch. Until 2015, TechCrunch was the owner of the Crunchbase data, but by that time Crunchbase decoupled itself from TechCrunch to focus on its own products. Crunchbase database contains up-to-date details about over 600000 companies, including a short description, a detailed description, number of employees, headquarters regions, contacts, market share, and the current areas of activity.

This paper compares different approaches for multilabel text classification, using recent information collected from Crunchbase. Each company is labeled with one or more categories, from a subset of 46 possible categories, and the proposed models predict the set of associated categories based solely on the company textual description. In order to address the multilabel problem, two classification strategies have been tested using different classification methods: a) we have created 46 binary models, one for each one of the categories, where the set of categories for a given description is achieved by combining the result of the 46 models; b) we have created a single model that gives the most probable categories for a given description. The resulting models may constitute an important asset for automatic classification of texts that can be applied, not only company descriptions, but to other texts, such as web pages, text blogs, news pages, etc. The work here described extends the work described in [2] to multilabel classification, and constitutes a more challenging task, since each record is associated with one or more categories.

This document is structured as follows: Sect. 2 overviews the related literature, focusing on the most commonly used methods and features to solve similar text classification problems. Section 3 describes the data extraction procedure, the resulting dataset, and the corresponding data pre-processing. Section 5 describes our experiments and the corresponding achieved results. Finally, Sect. 6 presents our final conclusions and pinpoints future research directions.

## 2 Related Work

Text based classification has become a major researching area, specially because it can be used for a large number of applications. The existing literature in text classification is vast, but most of the studies consider only a small number of possible categories.

Most text classification approaches are based on Supervised Learning. [14] applied machine learning to classify movie reviews from Internet Movie Database

(IMDb) by sentiment. Also in [9] an experiment to spam detection in customer reviews took place to check if false opinions were given to a product. Text classification has also been extensively applied to social media. The work described in [11] applies several algorithms to tweets trying to find “trending topics”, and [22] used Twitter information to develop an automated detection model to find rumors and misinformation in social media, achieving an accuracy of about 91%. These are examples of binary classification problems, but a bigger challenge arises when it comes to multiple categories, also known as multi-class. The work described in [3] presents a strategy based on binary maximum entropy classifiers for automatic sentiment analysis and topic classification over Spanish Twitter data. Both tasks involve multiple classes, and each tweet may be associated with multiple topics. Different configurations have been explored for both tasks, leading to the use of cascades of binary classifiers for sentiment analysis and a one-vs-all strategy for topic classification, where the most probable topics for each tweet were selected.

The performance and overall simplicity of Naive Bayes makes it a very attractive alternative for several classification tasks [13]. [8] used a Naive Bayes Classifier for author attribution applied to a dataset called AAAT dataset (i.e. Authorship attribution of Ancient Arabic Texts) obtaining results up to 96% classification accuracy. Naive Bayes results are mainly obtained from an unreal assumption of independence. For this, there has been a major focus on investigating the algorithm itself. Recently, [23] used a Naive Bayes on 20 newsgroups, and compared the Multinomial, Bernoulli and Gaussian variants of Naive Bayes approaches.

The work described by [7] reports the use of Fuzzy Fingerprints to find an author of a text document using a large Dataset of newspaper articles from more than 80 distinct authors, achieving about 60% accuracy. Also [18] and [5] make use of the same technique to solve a multi-class classification problem when trying to find events and twitter topics using textual data.

The work reported by [19] demonstrates that the use of Support Vector Machines (SVM) outperform many other methods when applied to text classification problems. The work described in [17] compares Naive Bayes and SVMs, using two well-known datasets with different sample sizes in multiple experiments, and concludes that SVMs outperform Naive Bayes by a large margin, giving a much lower error rate, at that time the lowest for the given sets of data. Also in [1] a text classification problem with a large number of categories is used to compare SVMs and Artificial Neural Networks (ANNs). The results are very clear for both recall and precision, both indicating the differences in performance of the SVM and ANN. The SVM once again outperforms ANN, suggesting that SVMs as more suitable for this type of problems, not only because they achieve better performance, but also because they are less computationally complex. Additionally, [1] also compares two sets of features, a large and a reduced feature set, concluding that, using SVMs, the small feature set achieves much better performance.

In what concerns features, one of the most common ways to represent textual data is the bag-of-words approach [6], a very simple and efficient way to quickly feed an algorithm and check its potential behavior. This method consists in a simple breakdown of a sentence into a set of words that are part of it. It usually achieves a decent performance, and in some cases, if the dataset is already very rich in terms of features it can be a good implementation. This type of approach assumes that words are independent, and do not consider the context where the word was used, losing the syntactic structure and semantic meaning of the sentence. When the data is sparse this technique may not be adequate, but it is possible to use a similar technique, based on n-grams, that preserves some of the local context of a word. The work reported by [10] describes experiments using n-grams (bag-of-n-grams), consisting in a n-size moving window (usually 1,2 or 3) along each sentence and collect the unique combination of words along with its count. Bag-of-words are compared to n-grams approaches and show a large improvement over the entire set of experiments. Another well-known and successful weighting scheme commonly used for Text Classification is Term Frequency - Inverse Document Frequency (TF-IDF). An alternative to bag of words, n-grams and TF-IDF is the use of word embeddings. Word embeddings are a much more complex technique that attempts to encode the semantics of the words in the text. Common implementations, such as word2vec, allow the use of embeddings in text classification often with good results. For example, [12] combines TF-IDF and word2vec and achieves more than 90% accuracy while processing a news dataset.

A set of Natural Language Processing (NLP) techniques are commonly used for extracting relevant features from a sentence. One of them is *lemmatization*, a way to prepare the text for further usage, and a widely used technique when working with text classifiers [15,21,24]. Unlike *stemming*, it is not just the process of removing the suffix of a word, it also considers the morphological structure of the word. *Stemming* is a similar approach that is much lighter to run, it does not look into the morphosyntactic form of a word [20]. *Part-of-Speech tagging* is another NLP technique commonly applied to text classification tasks, that consists of assigning a part-of-speech tag (e.g., noun, adjective, verb, adverb, preposition, conjunction, pronoun, etc.) to each word. For example, [16] use part-of-speech to approach a multi-class classification problem for Amazon product reviews.

An attempt to extract to automatically extract information from an older version Crunchbase has been done in [2]. At the time, Crunchbase contained around 120K companies, each classified to one out of 42 possible categories. The dataset also contained category “*Other*”, that grouped a vast number of other categories. The paper performs experiments using SVMs, Naive Bayes, TF-IDF, and Fuzzy fingerprints. To our knowledge, no other works have reported text classification tasks over a Crunchbase dataset.

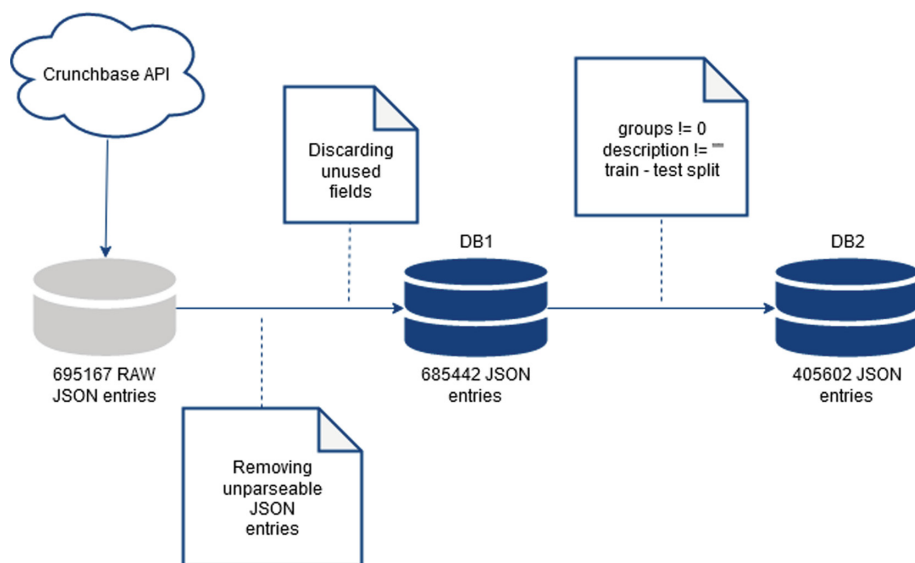


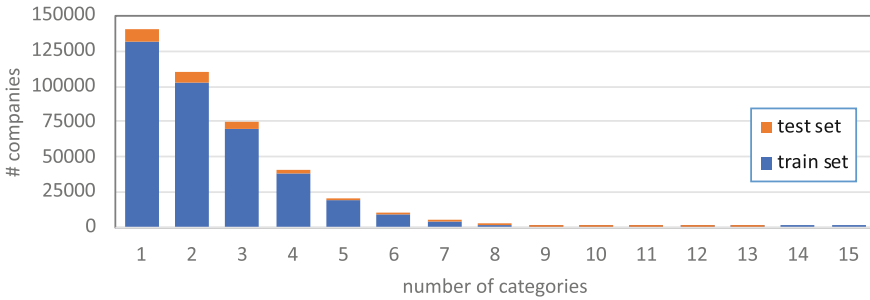
Fig. 1. Filtering and data transformation diagram.

### 3 Corpus

In this work we use a subset of a dataset extracted from Crunchbase containing up-to-date information of about 685000 companies. Crunchbase exposes a REST API that can be used to integrate all the information that is provided at Crunchbase for external applications. This API is accessible for researching purposes and Crunchbase Team, upon request. Crunchbase kindly provided full access for their data for 6 months, while developing this work. The data was extracted using the Crunchbase API and stored into SQLite3 databases. Figure 1 presents the data retrieval and filtering procedures. During the data retrieval stage we verified that some of the original responses were not retrieved properly, so they were removed. The extracted JSON entries contain a lot of information, but only a small portion of that information is relevant for our task: URL of the company, company name, description, short\_description, categories, and fine categories. DB1 contains only parseable records, containing only the relevant fields for our task. Finally, DB2 contains the data used in our experiments, were entries that did not belong to any category or that did not contain any description were filtered out. The final database contains a total of 405602 records, that have been randomly shuffled and stored into two different tables: *train*, containing 380602 records, will be used from training our models; and *test*, containing 25000 records, that will be used for evaluating our models.

Each record contains a textual description of the company, that explains the company for whoever wants to have a brief notion of what it does and the areas that it belongs, and a short description, which is a summary of the

description itself. Typically the textual description consists of only a couple of paragraphs, an average of 77 words (including stopwords) which makes the text based classification task a difficult problem. Crunchbase considers a fixed set of 46 distinct categories, that correspond to areas of activity, and labels each company with one or more of those categories. Figure 2 shows an histogram of the number of companies that were labeled with a given number of different categories. Most of the companies were labeled with more than one label, and the maximum number of categories is 15. The average number of categories for each company is 2.41, which may be a relevant fact to be considered in the evaluation stage.



**Fig. 2.** Histogram of the number of companies labeled with a given number of categories.

Each category can also be decomposed into a number of fixed *fine categories*. The category is wider (e.g. *Software*) while the *fine categories* are more specific (e.g., *Augmented Reality*, *Internet*, *Software*, *Video Games*, *Virtual Reality*). Each *fine category* can be present in more than one category, for instance “*Alumni*” appears as a *fine category* for “*Internet Services*”, “*Community and lifestyle*”, “*Software*”, and many other categories. Also, “*Consumer*” appears in “*Administrative Services*”, “*Hardware*” and “*Real Estate*”, among others. The analysis performed in this paper considers only the 46 wider categories.

Figure 3 presents the number of companies that have been labeled with a given group, revealing a highly unbalanced dataset, where the frequency of each category ranges from 28% (*Software*) to 0.7% (*Navigation and Mapping*). The “*Software*” category is assigned to over 100K records, while 17 categories occur in less than 10K companies. It is also important to note that even the second most represented category, *Internet Services*, corresponds to only 56% of the most represented category.

## 4 Experimental Setup

This section presents the details about the experimental setup. It starts by describing the corpora pre-processing steps, then it presents the adopted classification methods, and finally presents the used evaluation metrics.

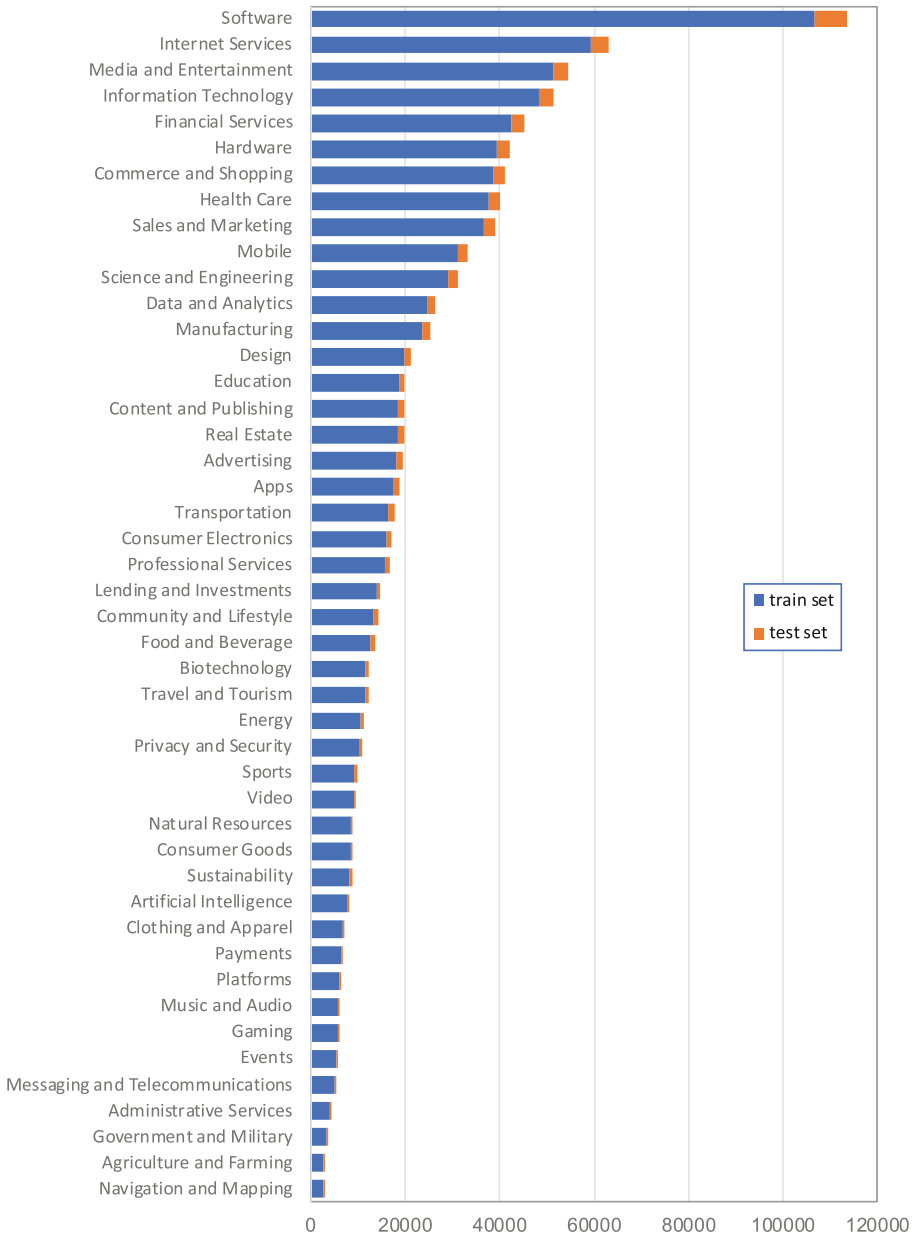


Fig. 3. Number of companies labeled with a given group.

## 4.1 Corpora Pre-processing and Feature Extraction

Experiments described in this paper model the categories of a company based on the text available in the company description. We have removed the punctuation marks, lower-cased every token, and removed all the stopwords from the text, by using the list of stopwords included with Natural Language Toolkit (NLTK) for the English language.

Concerning the text representation features, our baseline experiments use a bag-of-words representation, considering the word frequency for each description. We have also tested the TF-IDF weighting scheme, word bigrams, lemmatization, stemming, and part-of-speech tags.

## 4.2 Methods

We have tested three approaches: multinomial Naive Bayes [13], Support Vector Machines [19], and Fuzzy Fingerprints [2]. We implemented the first two approaches using scikit-learn, and used our own implementation of Fuzzy Fingerprints.

Naive Bayes is one of the most widely used methods for binary and multiclass text classification. The method outputs a probabilistic distribution, which makes it possible to analyse the probability of the outcome and to easily define thresholds. The multinomial Naive Bayes classifier is suitable for classification with discrete features, which is the case of word counts for text classification, given that the multinomial distribution normally requires integer feature counts. In practice, fractional counts such as TF-IDF may also work, but our experiments using TF-IDF achieved much worse performances.

Support Vector Machines (SVM) were introduced as a solution for a binary problem with two categories associated with pattern recognition. The SVM calculates the best decision limit between different vectors, each belonging to a category. Based on the limit minimization principle [4] for a given vector space where the goal is to find the decision boundary that splits the different classes or categories. SVM based models are often used in text classification problems since they behave quite well when used in supervised learning problems. The good results are due to the high generalization capacity of the method, which can be particularly interesting when trying to solve problems in bigger dimensions. Every experiment here described use the default parameters of the scikit-learn implementations.

Fuzzy fingerprints experiments use the *Pareto* function with  $K = 4000$ . For more information about the method refer to [2].

## 4.3 Evaluation Metrics

Experiments in this paper are evaluated using the metrics: accuracy, precision, recall and F1-score, defined as:

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ predictions}$$



$$\textit{Precision} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false positives}}$$

$$\textit{Recall} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}}$$

$$\textit{F1 - score} = \frac{2 * \textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}}$$

In order to calculate the overall metrics, we must consider the micro-average and macro-average versions of the above performance metrics. A macro-average computes the metric independently for each category and then takes the average (hence treating all categories equally), whereas a micro-average aggregates the contributions of all categories to compute the average metric. Micro-average metrics are usually preferable metric for unbalanced multi-class classification tasks.

## 5 Experiments and Results

This section presents two sets of experiments. Section 5.1 describes experiments with binary classification models, one for each category, where each model predicts a category. Section 5.2 presents a number of experiments, considering only one model in a multi-class scenario.

### 5.1 Binary Classification Models

Our first set of experiments consists of creating a model for each one of the categories. In order to train each model in a binary fashion, we have selected every companies labeled with the corresponding label as positive samples, and all the other companies as negative samples. In this scenario, the performance of each model can be evaluated individually, but micro-average or macro-average metrics must be used in order to assess the global performance.

Table 1 presents the most relevant micro-average results. Concerning the multinomial Naive Bayes, the best results were achieved using the word frequency, as expected (see Sect. 4.2). The performance of the SVM-based models improved when moving from the word frequency to the TF-IDF weighting scheme. However, the performance did not improve after introducing other NLP-based features, such as lemmatization, stemming, part-of-speech tags, or bigrams. The Fuzzy fingerprints did not produce interesting results, but this was an expected result due to the small size of the descriptions and the fact that they were developed for multi-class problems, and usually only are advantageous when dealing with large number of classes [7, 18]. Overall, the best result, considering the F1-score, was achieved by combining TF-IDF with SVMs.

Table 2 shows the performance of each individual classifier, revealing that the performance does not necessarily degrades for the most highly unbalanced categories.

**Table 1.** Classification results for each method, using different features.

Experiments	Accuracy	Precision	Recall	F1-score
Multinomial Naive Bayes (word frequency)	0.951	0.547	<b>0.439</b>	<b>0.487</b>
SVM				
Word frequency	0.950	0.537	0.412	0.467
TF-IDF weights	0.959	0.696	<b>0.420</b>	<b>0.524</b>
TF-IDF + Lemmas	0.960	0.702	0.416	0.522
TF-IDF + Stemming	0.959	<b>0.704</b>	0.410	0.518
TF-IDF + POS tags	0.959	0.701	0.405	0.513
TF-IDF + word bigrams	0.960	0.703	0.417	0.523
TF-IDF + word bigrams + POS tags	0.959	0.701	0.405	0.513
Fuzzy fingerprints (Word frequency)	–	0.204	<b>0.786</b>	0.324

The achieved results can not be directly compared with the results described in [2], not only because the evaluation sets differ, but also because different metrics and different modeling approaches are being used. However, it is interesting to note that SVMs, which did not perform well in that work, are now the best performing method (by far).

## 5.2 Multi-class Classification

Our second set of experiments consists of creating a single model that is able to provide the most probable category. In order to train such a model, we have duplicated each entry as many times as the number of corresponding category labels. So, each company labeled with  $n$  categories was duplicated once for each one of the categories, and used for training each individual category. Such a model may be useful for automatically guessing the best category for a given text and also provide the top best categories. The performance of the model cannot be easily evaluated, once the number of possible categories varies for each company. So, we have evaluated the performance of correctly predicting one of the categories, which may correspond to the best category only. In this scenario, Accuracy becomes the most adequate metric, and Precision and Recall do not apply. For this experiment we used TF-ICF (Term Frequency - Inverse Class Frequency) [18], a process similar to TF-IDF, in the Fuzzy Fingerprints approach.

Table 3 shows the classification performance when predicting the top category, for each one of the three methods. Our baseline corresponds to always guessing the most frequent category. In this multi-class experiment SVM and Fuzzy Fingerprints perform very similarly, but the SVM is around 16x slower. The multinomial Naive Bayes runs very fast, but the performance is more than 3% (absolute) lower than the other two approaches.

**Table 2.** Classification performance by group for SVM + TF-IDF

	Samples	Precision	Recall	F1-score
Software	6929	0.683	0.560	0.616
Internet Services	3956	0.584	0.297	0.393
Media and Entertainment	3338	0.710	0.485	0.576
Information Technology	3108	0.565	0.263	0.359
Financial Services	2767	0.826	0.670	0.740
Hardware	2630	0.658	0.349	0.456
Commerce and Shopping	2527	0.676	0.394	0.498
Health Care	2521	0.841	0.704	0.767
Sales and Marketing	2387	0.732	0.457	0.563
Mobile	2017	0.582	0.319	0.412
Science and Engineering	1949	0.731	0.428	0.540
Data and Analytics	1595	0.629	0.279	0.386
Manufacturing	1576	0.656	0.473	0.550
Design	1305	0.630	0.274	0.381
Education	1226	0.804	0.591	0.681
Content and Publishing	1233	0.643	0.354	0.456
Real Estate	1231	0.770	0.532	0.629
Advertising	1156	0.678	0.396	0.500
Apps	1190	0.449	0.100	0.164
Transportation	1155	0.749	0.435	0.551
Consumer Electronics	1084	0.534	0.122	0.198
Professional Services	1018	0.679	0.302	0.418
Lending and Investments	933	0.639	0.424	0.510
Community and Lifestyle	888	0.549	0.114	0.188
Food and Beverage	844	0.772	0.610	0.682
Biotechnology	766	0.747	0.560	0.640
Travel and Tourism	723	0.791	0.488	0.604
Energy	754	0.775	0.580	0.663
Privacy and Security	666	0.726	0.362	0.483
Sports	607	0.735	0.448	0.557
Video	563	0.648	0.393	0.489
Natural Resources	579	0.717	0.522	0.604
Consumer Goods	571	0.656	0.294	0.406
Sustainability	574	0.680	0.389	0.494
Artificial Intelligence	509	0.742	0.316	0.444
Clothing and Apparel	470	0.769	0.496	0.603
Payments	409	0.640	0.347	0.450
Platforms	375	0.429	0.048	0.086
Music and Audio	403	0.784	0.496	0.608
Gaming	358	0.651	0.453	0.534
Events	367	0.671	0.294	0.409
Messaging and Telecommunications	313	0.539	0.220	0.313
Administrative Services	272	0.574	0.129	0.210
Government and Military	220	0.511	0.109	0.180
Agriculture and Farming	222	0.725	0.392	0.509
Navigation and Mapping	173	0.476	0.116	0.186

**Table 3.** Multi-class classification results.

	Accuracy	Execution time (s)
Most frequent category (Baseline)	0.280	
Multinomial Naive Bayes	0.646	11.37
SVM	0.678	1010.35
Fuzzy Fingerprints	0.672	62.99

## 6 Conclusions and Future Work

This paper describes multi-label text classification experiments over a dataset containing more than 400000 records about companies extracted from Crunchbase. We have performed experiments using three classification approaches, multinomial Naive Bayes, SVM, and Fuzzy fingerprints, and considering different combinations of text representation features. Our dataset is highly unbalanced since the frequency of each category ranges from 28% to 0.7%. Nevertheless, our findings reveal that the description text of each company contains features that allow to predict its area of activity, expressed by its corresponding categories, with about an overall performance of 70% precision, and 42% recall. When using a multi-class approach, the accuracy for predicting the most probable category is above 65%.

We are planning to improve this work by considering additional evaluation metrics for ranking problems, such as precision@k, recall@k and f1@k, that may be suitable for measuring the multi-label performance. Additionally, we are also planning to introduce features based on embeddings and to compare the reported methods with other neural network classification approaches.

## References

1. Basu, A., Walters, C., Shepherd, M.: Support vector machines for text categorization. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003, pp. 1–7 (2003). <https://doi.org/10.1109/HICSS.2003.1174243>
2. Batista, F., Carvalho, J.P.: Text based classification of companies in CrunchBase. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–7 (2013). <https://doi.org/10.1109/FUZZ-IEEE.2015.7337892>
3. Batista, F., Ribeiro, R.: Sentiment analysis and topic classification based on binary maximum entropy classifiers. *Procesamiento de Lenguaje Nat.* **50**, 77–84 (2013). <http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/4662>
4. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995). <https://doi.org/10.1007/BF00994018>
5. Czarnowski, I., Jędrzejowicz, P.: An approach to rbf initialization with feature selection. In: Angelov, P., et al. (eds.) *Intelligent Systems 2014. AISC*, vol. 322, pp. 671–682. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-11313-5\\_59](https://doi.org/10.1007/978-3-319-11313-5_59)
6. Harris, Z.S.: Distributional structure. *Word* **10**(2–3), 146–162 (1954)

7. Homem, N., Carvalho, J.P.: Authorship identification and author fuzzy “fingerprints”. In: Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS, pp. 180–185 (2011). <https://doi.org/10.1109/NAFIPS.2011.5751998>
8. Howedi, F., Mohd, M.: Text classification for authorship attribution using naive bayes classifier with limited training data. *Comput. Eng. Intell. Syst.* **5**(4), 48–56 (2014). <http://iiste.org/Journals/index.php/CEIS/article/view/12132>
9. Jindal, N., Liu, B.: Review spam detection. In: Proceedings of the 16th International Conference on World Wide Web, pp. 1189–1190 (2007)
10. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint [arXiv:1607.01759](https://arxiv.org/abs/1607.01759) (2016)
11. Lee, K., Palsetia, D., Narayanan, R., Patwary, M.M.A., Agrawal, A., Choudhary, A.: Twitter trending topic classification. In: 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 251–258. IEEE (2011)
12. Lilleberg, J., Zhu, Y., Zhang, Y.: Support vector machines and Word2vec for text classification with semantic features. In: Proceedings of 2015 IEEE 14th International Conference on Cognitive Informatics and Cognitive Computing, ICCI\*CC 2015, pp. 136–140 (2015). <https://doi.org/10.1109/ICCI-CC.2015.7259377>
13. Murphy, K.P., et al.: Naive bayes classifiers. *Univ. Br. Columbia* **18**, 60 (2006)
14. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10, pp. 79–86. ACL (2002)
15. Plisson, J., Lavrac, N., Mladenic, D., et al.: A rule based approach to word lemmatization. In: Proceedings of IS, vol. 3, pp. 83–86 (2004)
16. Prankevicius, T., Marcinkevicius, V.: Application of logistic regression with part-of-the-speech tagging for multi-class text classification. In: 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering, AIEEE 2016 - Proceedings, pp. 1–5 (2017). <https://doi.org/10.1109/AIEEE.2016.7821805>
17. Rennie, J.D.M., Rifkin, R.: Improving multiclass text classification with the support vector machine. Technical report, October 2001, Massachusetts Institute of Technology AI Memo 2001–026 (2001). <http://dspace.mit.edu/handle/1721.1/7241>
18. Rosa, H., Batista, F., Carvalho, J.P.: Twitter topic fuzzy fingerprints. In: WCCI2014, FUZZ-IEEE, 2014 IEEE World Congress on Computational Intelligence, International Conference on Fuzzy Systems, pp. 776–783. IEEE Xplorer, Beijing, July 2014
19. Sain, S.R., Vapnik, V.N.: The Nature of Statistical Learning Theory, vol. 38. Springer, Heidelberg (2006). <https://doi.org/10.2307/1271324>
20. Sharma, D., Cse, M.: Stemming algorithms: a comparative study and their analysis. *Int. J. Appl. Inf. Syst.* **4**(3), 7–12 (2012)
21. Toman, M., Tesar, R., Jezek, K.: Influence of word normalization on text classification. In: Proceedings of InSciT, pp. 354–358 (2006). <http://www.kiv.zcu.cz/research/groups/text/publications/inscit20060710.pdf>
22. Vosoughi, S.: Automatic detection and verification of rumors on Twitter. Ph.D. thesis, Massachusetts Institute of Technology (2015)
23. Xu, S.: Bayesian naive bayes classifiers to text classification. *J. Inf. Sci.* **44**(1), 48–59 (2018)
24. Zhang, D., Chen, X., Lee, W.S.: Text classification with kernels on the multinomial manifold. In: SIGIR 2005–28th Conference on Research and Development in Information Retrieval, pp. 266–273 (2005). <https://doi.org/10.1145/1076034.1076081>