

UniversidadeVigo

STRATEGIES FOR THE PROGRAMMATIC  
GENERATION OF LABELLED CORPUS FOR  
TEXT CLASSIFICATION

Pedro Alonso Doval

Master's Thesis presented to the  
Telecommunications Engineering School  
Master's Degree in Telecommunications Engineering

Supervisors  
Juan Manuel Santos Gago  
Héctor Cerezo Costas

2021





# Abstract

This master thesis explores weak supervision technologies, focusing on the more innovative data programming strategies for automatically labelling large datasets of texts for complex problems in which human labelling is impractical. The objective of these techniques is to obtain a training corpus that can be used later for training machine learning models in text classification problems achieving comparable results to fully supervised models. To test this approach we selected a real problem: categorizing journalistic news into multiple predefined categories. This document presents the state of the art about data programming techniques, and describes the process of designing a solution using the Snorkel framework. The tasks covered by this master thesis include, first of all, the design of multiple label functions that serve as weak supervision sources, using an heterogeneous set of techniques such as heuristics or keyword detection, linear regression classifiers, clustering methods or deep learning models. Afterwards, the labels of the different sources are aggregated using a generative model, obtaining probabilistic labels for the news. Lastly, the final data set is obtained from the probabilistic labels. There is a performance analysis for different configurations using a crowdsourced dataset as reference. These results are evaluated to find out the potential applications of data programming techniques and how to take full advantage of them in practical scenarios.

Keywords:

Natural language processing

Weak Supervision

Data programming

Text classification

Label function



# Contents

<b>Abstract</b>	<b>i</b>
<b>List of figures</b>	<b>v</b>
<b>List of tables</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objectives . . . . .	4
1.3 Methodology . . . . .	5
<b>2 State of the art</b>	<b>7</b>
<b>3 Definition of the problem and evaluation strategy</b>	<b>11</b>
3.1 Definition of the problem . . . . .	11
3.1.1 Crowdsourcing strategy . . . . .	13
3.1.2 Test set creation . . . . .	15
<b>4 Design and implementation</b>	<b>19</b>
4.1 Selection of the framework . . . . .	19
4.1.1 Label functions . . . . .	21
4.2 Design and implementation of the solution . . . . .	22
4.2.1 Definition of the input format . . . . .	22
4.2.2 Introduction to useful concepts and tools . . . . .	22
4.2.3 Proposal of label functions . . . . .	24
4.3 Snorkel framework implementation . . . . .	35
<b>5 Results and evaluation</b>	<b>37</b>
5.1 Evaluation of label functions . . . . .	37
5.1.1 Results of individual label functions . . . . .	37
5.1.2 Statistics of label functions in the full dataset . . . . .	46
5.2 Evaluation of aggregation of label functions . . . . .	49
5.2.1 Majority vote model . . . . .	49
5.2.2 Full generative model . . . . .	51

## Contents

---

5.2.3	Final train set . . . . .	53
5.3	Ablation study . . . . .	56
5.3.1	Generative model excluding BERT LF . . . . .	56
5.3.2	Generative model excluding entities LF . . . . .	56
5.3.3	Generative model excluding monetarian LF . . . . .	57
5.4	Recapitulation of results . . . . .	57
<b>6</b>	<b>Conclusions and future work</b>	<b>59</b>
6.1	Conclusions . . . . .	59
6.2	Limitations . . . . .	61
6.3	Ethics . . . . .	61
6.4	Future work . . . . .	62
<b>7</b>	<b>Annex I - Results of ablation study</b>	<b>65</b>
	<b>Bibliography</b>	<b>72</b>

# List of Figures

1.1	Performance of classic machine learning and deep learning algorithms vs amount of training data . . . . .	2
1.2	Diagram with the different solutions to label data . . . . .	3
2.1	Difference between generative and discriminative models. . . . .	8
2.2	Diagram of the Snuba workflow. . . . .	8
2.3	Example of creation of LFs in Babble Labble . . . . .	9
3.1	Diagram with the problem to solve in this work. . . . .	12
3.2	Example of question in the crowdsourcing form. . . . .	14
3.3	Plot analysing the answers of the crowdsourcing form . . . . .	15
3.4	Plot analysing the answers of the crowdsourcing form . . . . .	16
3.5	Fraction of answers corresponding to the most voted label for each news article in the . . . . .	16
3.6	Distribution of received labels in the form . . . . .	17
3.7	Distribution of labels in the test set . . . . .	17
3.8	Conflicts between categories in the received labels . . . . .	18
4.1	Overview of the Snorkel system . . . . .	19
4.2	Advantage of models using data generated with weak supervision . . . . .	20
4.3	JSON format of the input dataset. . . . .	22
4.4	Diagram with the keywords based LF. . . . .	26
4.5	Diagram with the LF based on sports professionals names. . . . .	27
4.6	Diagram with the LF based on monetarian expressions. . . . .	27
4.7	Diagram with the workflow of the TF-IDF, logistic regression based LF. . . . .	28
4.8	Diagram with the workflow of the BERT based LF. . . . .	30
4.9	Diagram with the workflow of the entity based LF. . . . .	31
4.10	Diagram with the LF based in professions names. . . . .	32
4.11	Diagram with the workflow of the LDA based LF. . . . .	33
4.12	Diagram with the implemented workflow for Snorkel. . . . .	35
5.1	Confusion matrix for LF_LDA . . . . .	40
5.2	Confusion matrix for LF_Entities . . . . .	41
5.3	Confusion matrix for LF_Professions . . . . .	42

## List of Figures

---

5.4	Confusion matrix for LF_BERT_1 . . . . .	44
5.5	Confusion matrix for LF_BERT_3 . . . . .	45
5.6	Conflicts among labels in the LFs outputs. . . . .	48
5.7	Histograms analyzing the output of the LFs. . . . .	49
5.8	Confusion matrix for the Snorkel results with a majority vote strategy . .	50
5.9	Confusion matrix for the Snorkel results with the generative model . . . .	51
5.10	Histogram of probabilities of the most probable label after the generative model . . . . .	52
5.11	Number of news of each category in the final train set . . . . .	53
5.12	Confusion matrix for the final train set results . . . . .	55
5.13	Comparative of accuracy results of different implementations. . . . .	58
7.1	Confusion matrix for the Snorkel results with a generative model excluding the BERT based LFs . . . . .	65
7.2	Confusion matrix for the Snorkel results with a generative model excluding the entities LF . . . . .	66
7.3	Value of probability of the most probable label for each record after excluding the Entities_LF . . . . .	67
7.4	Confusion matrix for the Snorkel results with a generative model excluding the monetarian LF . . . . .	68



# List of Tables

4.1	Cross validation metrics for different ratio of labels in the TF-IDF based LF	29
4.2	Table with the different prompts used to test BERT based LF.	30
5.1	Metrics of a random baseline classifier	38
5.2	Metrics of the keywords LFs	38
5.3	Metrics of the TF-IDF LFs	39
5.4	Metrics of LF_LDA	39
5.5	Metrics of LF_Entities	41
5.6	Metrics of LF_Professions	43
5.7	Metrics of LF_BERT_1	43
5.8	Metrics of LF_BERT_3	44
5.9	Statistics of the label functions in full dataset	47
5.10	Metrics for the Snorkel results with a majority vote strategy	50
5.11	Metrics for the Snorkel results with the generative model	52
5.12	Examples of misclassified news in the final train set	54
5.13	Metrics for the final train set results	55
7.1	Metrics for the Snorkel results with a generative model excluding the BERT based LFs	66
7.2	Metrics for the Snorkel results with a generative model excluding the entities LF	67
7.3	Metrics for the Snorkel results with a generative model excluding the monetarian LF	68



# 1 Introduction

## 1.1 Motivation

Natural language processing (NLP) is a field of computer science, artificial intelligence and linguistics that studies the understanding, interpretation and handling of the human language by machines.

One of the main applications of NLP is text classification. A text classification problem consists of assigning a textual input to a determined category or label. In a classification problem the possible labels are known and discrete. The majority of classification systems are based on supervised machine learning (ML). In supervised ML systems the classification is based on knowledge acquired from past observations. For these systems to work, they need a group of labelled texts as input for learning to match the different labels to the characteristics of the texts. This learning process is known as training and it generates a model as a result. The model consists of the set of parameters or characteristics which define the behaviour of the classifier for an specific algorithm and contains the knowledge acquired during training.

Different problems and algorithms need a different number of labelled texts for training to achieve high accuracy in their predictions, varying from hundreds to even millions of texts.

Among the most popular algorithms for supervised ML, we find algorithms such as Support Vector Machine (SVM) [1], Naive Bayes [2] or Logistic Regressions [3]. Another group of algorithms which increased popularity and development during the last years are Deep Learning algorithms. They are based on structures called artificial neural networks [4] inspired on how the human brain works. One characteristic of Deep Learning systems is that they require a big quantity of training data and that their performance usually improves with the number of training data, in opposition to non Deep Learning algorithms which usually have a threshold that they can not exceed by adding more training data

[5]. This phenomenon is represented in the plot of the figure 1.1.

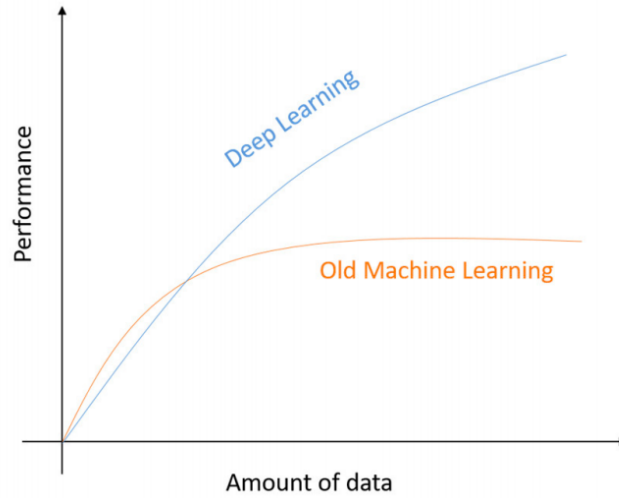


Figure 1.1: Performance of classic machine learning and deep learning algorithms vs amount of training data. Source: [5].

Additionally to supervised ML, there is also unsupervised ML, consisting principally of clustering methods, where the system does not have predefined outputs. In this case it just discovers from the input samples their characteristics and can group them by similarities or by characteristics they share, but these groups are not previously established. Moreover, semi-supervised ML is an intermediate approximation that combines a small amount of labelled data with a large amount of unlabelled data during training.

The lack of labelled data for model training is a recurring problem in the domain of supervised ML. Obtaining an appropriate set of labelled data is generally costly, both in time (large amounts of data are often required for effective training) and personnel (labelling often involves highly specialized domain experts). Such is the case that in many real problems it is not feasible to obtain these data sets in a reasonable time. Thus, the ML community has generated attention in developing strategies to efficiently obtain labelled data from a (potentially very large) unlabelled data set.

There are different ways of obtaining labelled data. Figure 1.2 explains them briefly. To begin with, the traditional supervision, where either the data are obtained already labelled from a trusted source, or the labels for a given unlabelled data set are obtained by means of the assignment by subject matter experts (SMEs): people who have high knowledge about the subject of the concerning data. This option can be out of reach due to the cost of labelling data by experts or the huge amount of data to label. Because of this issue, sometimes it can be appropriate resort to weak supervision approaches. Weak supervision strategies involve techniques that allow to obtain the labelled training data without manual labelling all the samples. These techniques include crowdsourcing (ask multiple people, possibly not as truthful as SMEs, to label small amounts of samples each

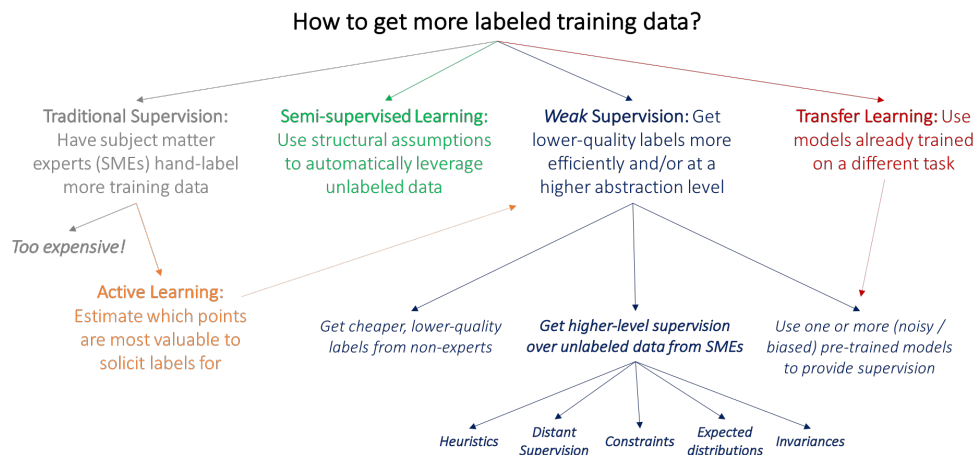


Figure 1.2: Diagram with the different solutions to label data [6]

one) or using existing knowledge resources (e.g. ontologies) about how to recognize and distinguish the samples of the different categories to automatically label the samples. In many cases, it produces lower-quality label data (compared to SMEs direct labelling) or labels at a higher abstraction level, but involve less cost and a more efficient use of the human effort available.

An intermediate solution to this is what is known as active learning: make use of SMEs more efficiently by having them label only data samples which are estimated to be most valuable to the model, and just using weak supervision strategies to the rest of samples, taking profit from the information from the labelled ones.

Another related strategy is transfer learning, consisting in leverage models or classifiers already trained for a task or dataset, and reusing them for another different but similar task, taking advantage from the knowledge it already has. A simple example to illustrate this can be the following practical scenario. Consider the case in which you have a model for sentiment analysis that classifies texts into positive or negative depending the feeling the writer transmits on it, and now you want a model for recognizing texts with hate speech. A good option here is to re-train the model starting from the parameters it currently has (this is called fine-tuning) and ‘teach’ it to recognize texts that contain hate speech, and it will need less training data than if it started learning from zero, because the model already had knowledge about the language and only needs to be taught about recognizing hate speech.

One of the most interesting and novel techniques of weak supervision is what is known as data programming [7]. In data programming users build several label functions (in advance, LFs) that solve totally or partially the problem. Those LFs are typically programmatic (e.g. rules or heuristics that combine different analysis) but they can also be previous outdated ML classifiers used for the same or similar tasks. Those functions

produce labelled data which is usually more imprecise than a fully supervised approach and even conflict among them (multiple LFs output different labels for the same input). Indeed, sometimes those LFs are incomplete: they can only produce a certain subset of labels. The hypothesis for using data programming is that by properly combining those labels from multiple functions the noise in the output will be reduced, giving as a result an appropriate set of labels that could form a training set and finally produce a trustful classification model for the addressed problem. As they are programmatic, the cost of generating the labels is low in comparison to manually tagging the data. It can be a cost efficient approach in many practical scenarios to ask SMEs to spend time defining a set of rules, heuristics, patterns and characteristics that can help to recognize the different categories and apply with them weak supervision strategies over unlabelled data to obtain a big, but probably noisy, corpus of labelled data useful for training a new model.

This master thesis was developed in collaboration with the Galician Research and Development Center in Advanced Telecommunications (Gradiant). Gradiant participates in projects where clients need to classify their textual data in several categories, which can change over time. It is not uncommon the absence of labelled data for such problems. Thus, the RD centre has interest in knowing and gain experience with techniques that allow it to obtain efficient and adaptable labelling systems with low supervision. Also, the NLP community deals with a bigger problem when working with other languages rather than English due to the small presence of these languages in academic corpus, studies and research in general.

Therefore, the focus of the work will be to explore the new data programming techniques to generate, from an unlabelled set of data, a large labelled corpus earmarked for being a training set for a specific text classification problem.

This problem will serve as playground with the objective of identifying the best strategies and ways of applying them to problems of labelling texts in Spanish, opening the door to its direct application when necessary in the future.

## 1.2 Objectives

The main objective of this master thesis is to identify a series of mechanisms to generate automatically labelled data for training ML systems centered in the field of text classification, observe their results and evaluate their efficiency, efficacy and utility. These mechanisms will be applied to a news categorization problem in which, from a set of unlabelled journalistic news, we want to obtain the correspondent labels from a defined taxonomy (e.g. sports, reports, etc.).

After finishing this process, the labelled data can be used to train text classification

systems. By studying the labels obtained we want to validate the utility of this kind of strategies (effort employed, final size of the labelled corpus obtained, etc.) and the quality of the obtained labels.

With the attainment of these objectives we want to obtain the necessary knowledge to understand the difficulties of applying these programmatic strategies for generating labelled data from unlabelled sources in real problems. The following questions should be answered:

- How difficult is to apply data programming to obtain a labelled corpus in real problems?
- How can these mechanisms be used to improve the existing models? For example, updating the training of a model when the context has changed from the original labels.
- What problems present this kind of approaches?

## 1.3 Methodology

To achieve the aforementioned objectives the following steps will be carried out:

1. Study of the state of the art. Research about the current state of the art looking for frameworks or techniques for doing this weak labelling, especially focusing on data programming.
2. Specification of the case of study: In order to test the strategies we will use a real text classification problem with some expected results. For this, it is necessary to establish different categories for labelling the set of texts and define an evaluation and validation strategy.
3. Selection of a framework. After analysing and evaluating the possibilities discovered in the state of the art, a framework or working strategy for making the programmatic labelling process will be selected according to the needs and the characteristics of the problem to solve.
4. Design and implementation of a set of weak labelling mechanisms. We will design a heterogeneous group of LFs for providing labels to the original dataset. This part requires certain research for obtaining a set of strategies that covers from simple methods based on heuristics, patterns or rules to more complex systems such deep learning models or clustering algorithms. We will also need to implement the workflow for obtaining the final results from the different label sources outputs. We will iterate in the development process: after some LFs are developed there

will be executed preliminary tests with a small subset of the dataset to verify the functioning of the framework.

5. Controlled execution of the labelling mechanisms and measurement of the results. First of all, the implemented process to obtain the labels for the selected dataset will be run and finally, using the previously defined validation strategy, we will obtain objective measurements of the results.
6. Analysis of the results. From the obtained metrics and measurements, we will analyze the strengths and weaknesses of the strategy implemented and we will compare the different solutions that have been developed. Finally we will summarize the conclusions about the fulfillment of the objectives of this thesis.



## 2 State of the art

This section addresses the different strategies, techniques and tools that have been developed by the academic community to generate training records with data programming.

One of the tools most mentioned and used in real cases is Snorkel [8]. It is a framework, available as a Python library, developed in 2019 originally by Stanford AI Lab for facilitating the labelling of samples without human intervention based on a combination of several noisy label sources, what is known as label functions (LFs). The LFs can be very heterogeneous. From simple heuristics that inspect the structure or content of the text, to classification models of increasing complexity. With the results of these multiple LFs, Snorkel applies a label aggregator, called generative model, to predict the probability for each sample of belonging to each category. Finally, these probabilistic labels can be used for training models for text classifications tasks. These models are called discriminative models.

Figure 2.1 helps introducing these two concepts. Generative algorithms model how data was generated for, based on its generation assumptions, estimate which category is most likely to generate the records. Meanwhile a discriminative algorithms does not care about how the data was generated, it simply learns how to categorize it.

Snorkel also includes procedures for data augmentation, i.e., the generation of new artificial samples to enlarge the size of a dataset. An interesting implementation of this is Snorkel Drybell [10], an extension of Snorkel made by Google to use them in real text classification problems. In this study Google compares, for some tasks, the performance achieved by a model trained with a training set obtained with Snorkel with the performance of the same model trained with different sizes of handlabelled data. It concludes that, as the labels have more noise, for an equivalent performance you need a bigger amount of Snorkel labelled data than handlabelled data, but obtaining the labels by this method has a much lower cost and is more time efficient than handlabel the necessary size of data.

Things have gone such well for Snorkel that it is now an independent company whose

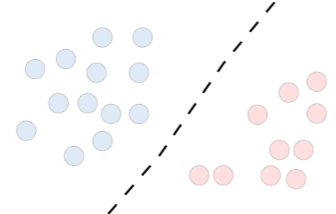
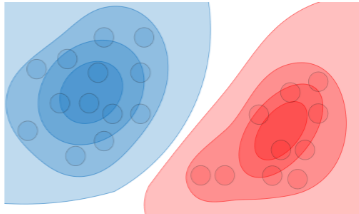
	Discriminative model	Generative model
Goal	Directly estimate $P(y x)$	Estimate $P(x y)$ to then deduce $P(y x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Figure 2.1: Difference between generative and discriminative models. Source: [9]

software is used by several reputed companies<sup>1</sup>.

Also Stanford AI Lab developed Snuba [11], a tool that extends Snorkel, that intends to solve the significant amount of time the experts spend designing the weak supervision sources making easier to adopt them. The main point of Snuba is the automatization of the generation of heuristics or LFs from a small labelled dataset. It is based on a three steps iterative process showed in figure 2.2.

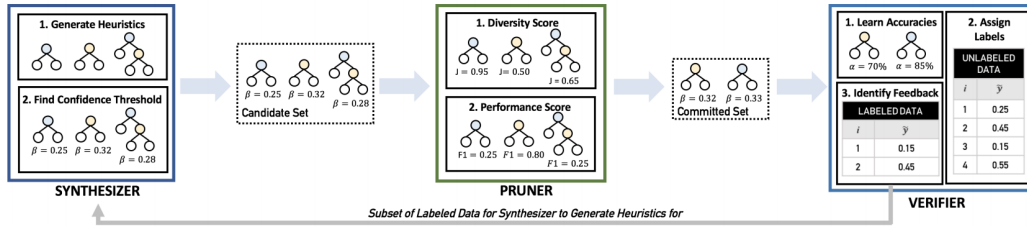


Figure 2.2: Diagram of the Snuba workflow. Source: [12]

The first step is the "synthesizer". It takes as input the small labelled dataset or, after the first iteration, a subset of the unlabelled dataset. It creates candidate heuristics that consist of, basically, classification models, such as logistic regressors, K-Nearest neighbour classifiers and decision stumps. These models assign probabilistic labels to the data points, that according to some thresholds, can be turned into actual labels, or result in "abstained" label.

Second step is the "pruner". It takes multiple candidate heuristics created by the synthesizer and selects the better for adding it to the existing set of heuristics to use. For ranking them, it focus on their performance over the labelled dataset but also on if they

<sup>1</sup><https://snorkel.ai/case-studies/>

---

label points that are not already labelled by other heuristics.

Finally, the "verifier" uses a generative model for producing one probabilistic training label for each point from the labels produced by the set of heuristics, and it passes to the synthesizer the points in the labelled dataset that have low confidence labels. The stopping condition is met if either there are no low confidence points in the labelled dataset or if the synthesizer is not learning properly the accuracies of the heuristics, and otherwise, another iteration starts.

Snuba seems to be an useful tool that can reduce the human effort for designing heuristics and LFs. According to its paper [11] it can outperform labels from user-defined heuristics and from semi-supervised learning. The downside is that, up to now, it is just focused on research, and lacks exhaustively tested and documented implementation like for example Snorkel already has.

Another available tool, also focused in research, is Babble Labble [13], again developed by Stanford. It agrees with Snorkel in obtaining multiple LFs and then applying an aggregator, but in this case the LFs are created by a parser. It needs an annotator to give the natural language explanation in some examples about why to give them a certain label, and converts to code that explanations, like in the example of figure 2.3. Babble Labble is an interesting proposal, but currently the parser works with a reduced grammar that can complicate generalising to diverse labelling tasks.

**Example**

Both cohorts showed signs of optic nerve toxicity due to ethambutol.

**Label**

Does this chemical cause this disease?

☒ Y ☐ N

**Explanation**

Why do you think so?

Because the words "due to" occur between the chemical and the disease.

**Labeling Function**

```
def lf(x):  
    return (1 if "due to" in between(x.chemical, x.disease)  
            else 0)
```

Figure 2.3: Example of creation of LFs in Babble Labble. Source: [13].

Flying Squid [14] is other framework with the same purpose. In this case, it has a lot of similarities with Snorkel. Users also have to write LFs, but differs in the label aggregator model, where FlyingSquid drastically reduces the time it takes for learning the accuracy and correlation parameters of LFs. The paper [14] states it can speed-up

even 440 times faster than previous data programming frameworks achieving comparable or higher performance.

Other tool that contains algorithms for automatic labelling is KNIME [15]. KNIME is a data mining platform with lots of functionalities in a visual environment, one of them is a workflow for weak supervision labelling. This workflow is similar to Snorkel, based on writing LFs and applying a generative and a discriminative model, but it is directly implemented in the visual environment. In this case, thanks to the interface, the users just have to focus in defining the data and the LFs.

Finally, some of the authors of Snorkel and Snuba proposed a new paradigm called Socratic learning [16]. It tries to solve some problems arising from assumptions that previous approaches make to simplify the process. Previous methods assumes that weak supervision sources have uniform accuracy over the entire datasets, which is rarely true. To solve this, it includes a cooperative dialog between the generative model and a discriminative model, that helps to automatically identify the subsets where the LFs perform worse and improve the performance over them.

To sum up this review of tools for weak supervision and data programming, we can appreciate, first of all, the big contribution to this field made by Stanford University. We can see all of the tools and frameworks discussed during this section make use of LFs, differing in the way of writing or obtaining them. These are new tools and strategies, as we can see all of them have been created or published in the last few years, so it is normal that some of them have just research purposes and therefore they lack some functionalities or the support suitable for a real use case.

## 3 Definition of the problem and evaluation strategy

### 3.1 Definition of the problem

For proving and evaluating the potential of data programming mechanisms for text classification, this work defines a specific problem to solve using these technologies. The problem to solve is obtaining a labelled training set from an unlabelled large dataset owned by Gradiant. This dataset is composed of 10 millions of unlabelled news articles in Spanish of the last years extracted from digital press services from all over the world. As the size of the dataset is so big, we will focus just in a subset of half million news. The original dataset consists of two files: one including one news's title per line, and the other containing one news's body per line. The lines in the same position of both files make up a title-body tuple of the same news.

A taxonomy has been defined for having a small set of categories to label the news. It does not cover all the dataset, but covers the biggest sections in journalistic news. It consists of 6 categories and takes into account the possibility of letting the system abstain, so the news not considered in any of the categories will not be included in the train set derived from this process. The categories are *sports*, *politics*, *economy*, *science and technology* (including news about science, technology, environmental issues or health related questions), *people* (news about famous people, celebrities, artists, television shows, etc.) and *reports* (incidents about crime, disasters, accidents, etc., or every surprising or curious fact happened in the society). For simplicity, in advance the category *science and technology* will be named just as *science*.

After the application of data programming techniques we will obtain a set of labels for some of the news, that will conform the train set that it can later be used for training classification models. The train set will exclude the news not belonging to a category in our taxonomy and the ones with low-confidence labels, only selecting the news with clear labels. Figure 3.1 shows a diagram of this process.

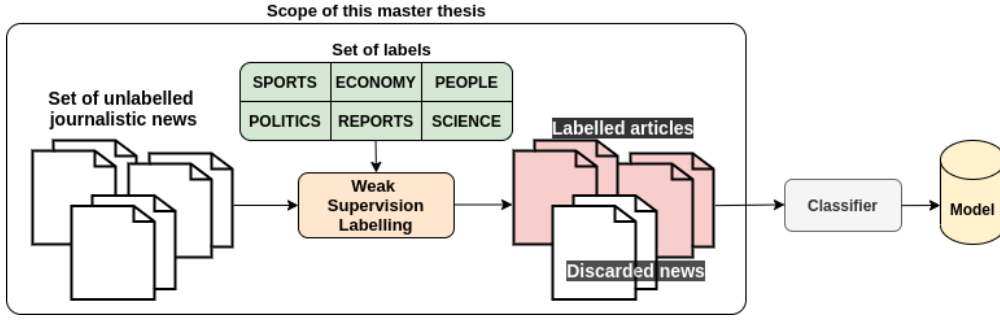


Figure 3.1: Diagram with the problem to solve in this work.

Once labelled the data we would want to validate these results. For this, there are some relevant variables: the effectiveness in the labelling (fraction of well classified news and size of the resultant training set) and efficiency (duration of the labelling execution and resources needed).

To evaluate the effectiveness it will be necessary to get a "ground truth" labelled subset (test set) and compare the results of the weak supervised labeling process with this test set using some predefined metrics. The strategy for creating this test set is selecting a random subset from the dataset and labelling it by crowdsourcing. This means, a group of diverse people will be asked to assign to the news the label they think is more suitable, according to the given definition of the categories, allowing them to abstain if they think the news do not belong to any of those categories. Each news will be labelled for various people, and in case of disagreement we implement a resolution strategy. This process is detailed in a further section.

The effectiveness metrics that will be used are:

- **Precision (P)** or positive predictive value. The precision for a category is the number of true positives, TP, (i.e. the number of news correctly labelled as belonging to such category) divided by the total number of news labelled as belonging to the category (i.e. the sum of true positives and false positives, FP, which are news incorrectly labelled as belonging to the category).

$$P = \frac{TP}{TP + FP}$$

- **Recall (R)** or sensitivity. The recall for a category is the number of TP divided by the total number of news belonging to that category (i.e. the sum of TP and false negatives, FN, which are news incorrectly labelled as not belonging to the category).

$$R = \frac{TP}{TP + FN}$$

- **F1-score.** F1-score can be interpreted as a metric that reflects a balance between precision and recall, so it is usually the one considered to evaluate how well performs the system for a category. It is calculated as the harmonic mean of them:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

- **Accuracy.** This is the fraction of news well classified in the whole test set, so it can give a general numerical evaluation of the whole system.

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

- **Macro average of precision, Macro average of recall, and Macro average of f1-score.** These measures the average value of one of those metrics between the values of all the categories, not taking into account the different number of examples of each category that could form the test set.
- **Macro weighted average of precision, Macro weighted average of recall, and Macro weighted average of f1-score.** Macro weighted average of the metrics is the average value but taking into account the distribution of the categories and weighting the values with the corresponding proportion.

Also it will be measured the average time it takes to run each LF and the time it takes to run the whole solution.

#### 3.1.1 Crowdsourcing strategy

To accomplish the crowdsourcing process for creating the test set for validation several options have been considered. The objective was to find the more suitable option that allows to handle the data in the easiest way and, at the same time, being simple and intuitive for the crowdsourcers.

One of the suggested options was using Quiztionary, an internal tool of Gradiant that allows to create questionnaires or surveys. This tool uses an API REST based on Loopback<sup>1</sup> for upload the data and get the results, and shows the questionnaire in a simple web interface. The drawbacks are that it would not be available for people outside Gradiant so the number of possible crowdsourcers would be reduced.

Another option that was considered, probably the simplest possible one in terms of creating the questionnaire, was to arrange the data to label in a CSV format so the crowdsourcers just have to open it with a spreadsheet program and write the label for

---

<sup>1</sup><https://loopback.io/>

### Chapter 3. Definition of the problem and evaluation strategy

---

each sample in the corresponding column. It is simple, but working directly with the data table is not very visually friendly for the crowdsourcers.

Finally, the best option considered was using Google Forms, setting each news as a question with multiple answers, with the categories being the set of answers. For arranging the data into the form, the Apps Script<sup>2</sup> Google's service offers an API that allows to interact with different Google services (Docs, Slides, Calendar, etc.) from JavaScript code. In this case, it was used for reading a spreadsheet in Google Sheets with the data to label in a CSV format and arrange it into multiple choice questions in a Google Form. This form with hundreds of questions was configured to show the questions in a different random order each time it is opened, so it is appropriate for allowing each crowdsourcer to answer only the firsts questions and ignore the rest of them, and by this way all the questions would be answered if the number of workers participating is high. The interface is as simple as shown in figure 3.2.

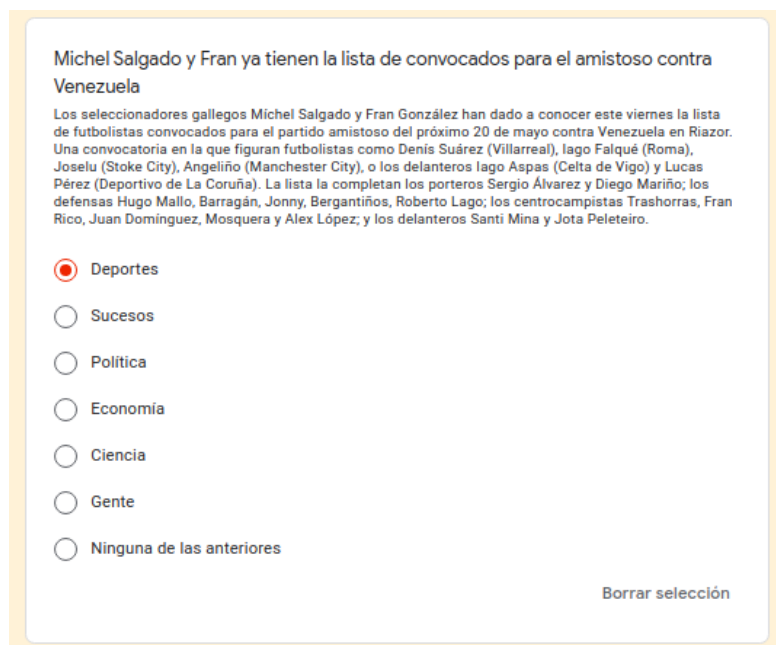
The image shows a Google Form interface. At the top, the title is "Michel Salgado y Fran ya tienen la lista de convocados para el amistoso contra Venezuela". Below the title is a paragraph of text in Spanish: "Los seleccionadores gallegos Michel Salgado y Fran González han dado a conocer este viernes la lista de futbolistas convocados para el partido amistoso del próximo 20 de mayo contra Venezuela en Riazor. Una convocatoria en la que figuran futbolistas como Denis Suárez (Villarreal), Iago Falqué (Roma), Joselu (Stoke City), Angeliño (Manchester City), o los delanteros Iago Aspas (Celta de Vigo) y Lucas Pérez (Deportivo de La Coruña). La lista la completan los porteros Sergio Álvarez y Diego Mariño; los defensas Hugo Mallo, Barragán, Jonny, Bergantiños, Roberto Lago; los centrocampistas Trashorras, Fran Rico, Juan Domínguez, Mosquera y Alex López; y los delanteros Santi Mina y Jota Peleteiro." Below the text is a list of radio button options: "Deportes" (selected), "Sucesos", "Política", "Economía", "Ciencia", "Gente", and "Ninguna de las anteriores". At the bottom right of the form is a button labeled "Borrar selección".

Figure 3.2: Example of question in the crowdsourcing form.

The instructions given to the volunteers for crowdsourcing were to expend the time they decide (ideally at least 5 minutes) labelling the news in the order they are shown. Pilot studies estimated that in that time can be labelled about 25 or 30 news. Obviously the texts are usually long, but the category of most of the news can be guessed just by reading the title, so this speeds up a lot the process.

Considering the number of people expected to participate in the labelling process, and the goal to have around 4 or 5 labels to each news, we calculated an optimal size for the

---

<sup>2</sup><https://developers.google.com/apps-script>



form of about 850 or 900 news. Due to the limit of 6 minutes for executing a script in the service Apps Script, the size was truncated in 873 news, and it was established as the final size.

After receiving all the answers, a spreadsheet with the results can be downloaded to proceed to analyze them and create the test set. The number of answers received in the form was 92, adding up a total of 5162 labels, which makes an average number of labels to each news of 5.9.

#### 3.1.2 Test set creation

For creating the test set from the answers received in the form, first of all is necessary to study and analyze the answers and labels received.

Figure 3.3 shows the number of answers that received the news. For example, we can see that there are 148 news that have been labelled 4 times. These values matches and even exceeds the expected, having for most of the news at least 4 labels.

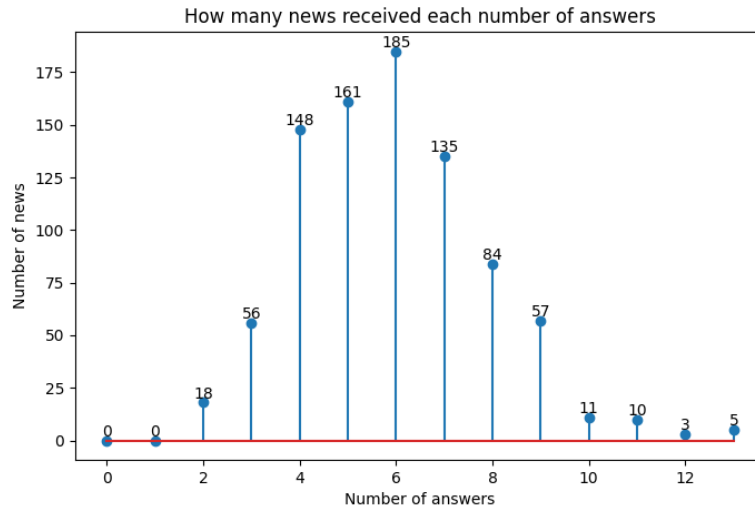


Figure 3.3: Number of news that received each number of labels.

Also we can see in figure 3.4 the number of different labels have been given to the news. There are 353 news that seem to appear pretty straightforward to categorize because they have been labelled always in the same category. Meanwhile in the rest there are discrepancies among the different answers. In order to analyze how strong are these discrepancies, figure 3.5 is an histogram in which each bin shows the number of news where the fraction of answers voting for the most voted label is in the interval of that bin, for example, the last bin corresponds to the number of news where the most voted label was selected close to the 100% of times, and we see the value agrees with the 353 of news

with just one different label in the answers seen in figure 3.4.

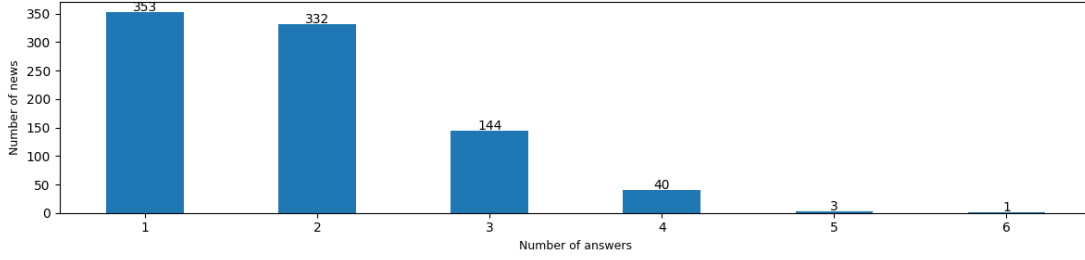


Figure 3.4: Number of news that received each number of different labels.

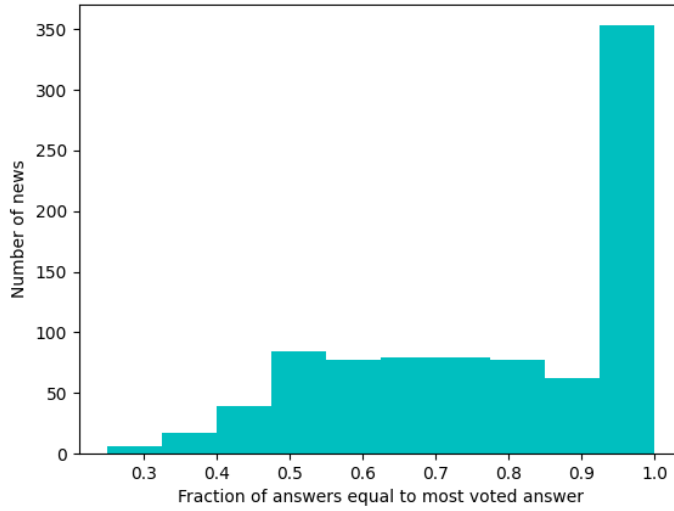


Figure 3.5: Fraction of answers corresponding to the most voted label for each news article.

Up to this point, we need an strategy to select which news are going to be part of the final test set and which ones will be discarded due to not being clear the category they belong to. Obviously the ones always labelled in one category must be included, and for the ones with conflict a threshold must be selected. This threshold has been selected to pick the ones having a majority strictly higher than  $2/3$ , estimating that this is a sufficient majority to consider the label as good.

If we go into the categories of the labels received, figure 3.6 shows the total number of labels received for each category. It can give a first approach of the distribution of the dataset. After the selection of the news that finally are part of the test set, the distribution is as we can see in figure 3.7, which is more or less correlated with the previous distribution. The test set counts with a total of 569 news.

### 3.1. Definition of the problem

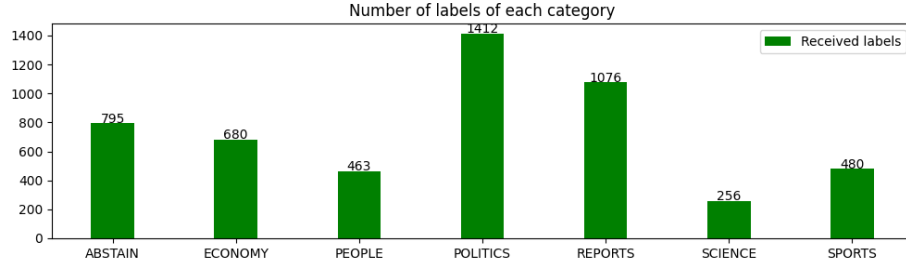


Figure 3.6: Distribution of received labels in the answers of the form

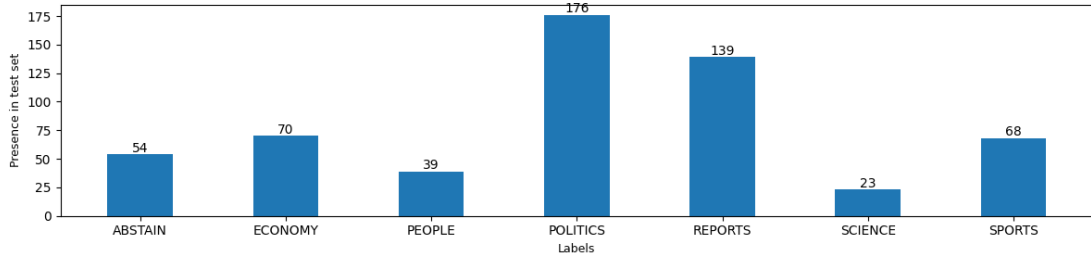


Figure 3.7: Distribution of labels in the test set.

Finally, figure 3.8 shows the overlapping over different categories in the answers received. For example, 148 news have been labelled both as *economy* and as *politics* in different answers (no matter if it has been 5 *politics* vs 1 *economy*, or 2 *politics* vs 2 *economy*). This is very useful to detect the nature of the dataset, where sometimes news can fit in more than one category, or where sometimes the category of some news is not clear even for human perception.

We can see that *economy* and *politics* are two categories that overlap more than others, which makes sense, because economical and political activities are usually very related. Between other categories this relation is softer. In the other hand, we can see that *sports* category is the one with less overlapping, so it means that usually sports news are the more evident to categorize. Finally, we can also appreciate that excepting for *sports*, for the rest or categories there is a relevant overlapping with abstention, so in many cases people do not agree about including or not a news article in a category.

The important point in this last analysis is that news in the dataset are not always evident to categorize (there is a lot of subjectivity), so the overlappings discovered must be taken into account when analyzing the performance results.

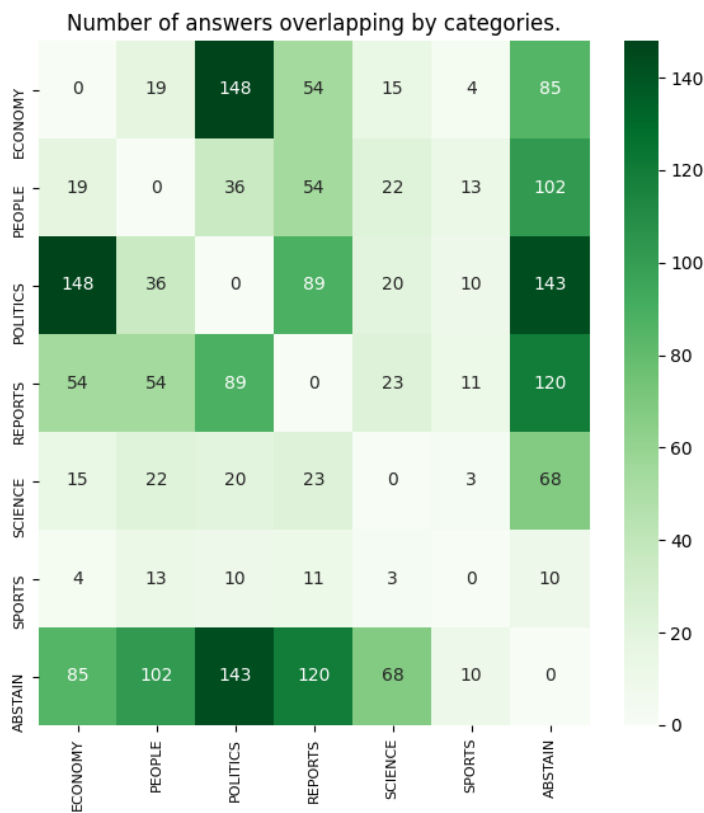


Figure 3.8: Conflicts between categories in the received labels

## 4 Design and implementation

### 4.1 Selection of the framework

Over all of the frameworks and tools for data programming studied in the state of the art section, the most mature, referenced and documented is clearly Snorkel. Thus it is the selected framework for the implementation.

Snorkel is a framework that can be used for other types of data like images, but our focus is in texts. The labelling process with Snorkel is a workflow that consists of several steps, represented in figure 4.1. First of all, multiple user-defined LFs allow to evaluate and label the samples. Each LF, following different criteria, will assign a label to each sample, or abstain if it does not have clear or enough information to decide.

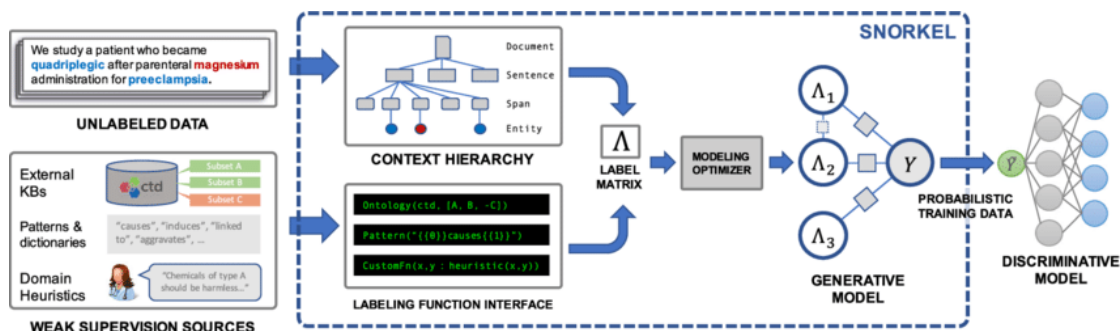


Figure 4.1: Overview of the Snorkel system. Source: [8]

Once all the outputs of the LFs are calculated for all samples, there are various strategies to achieve the final labels. The simplest one is a majority vote strategy: the label most repeated among the LFs for each sample is selected. The other is training a generative model. This is a model that will learn about the correlations and dependencies in the labels predicted by the different functions and therefore define different weights for each of them. Thus, from outputs of the LFs for each sample, it estimates the probabilities

of belonging to each of the categories. Finally, with the probabilistic labels we have to make up the final labelled set, by different strategies, depending on their interests: for example collecting just the texts that have labels above a threshold probability, or just maintaining the probabilistic values if the target application requires it.

Up to this point, one can think *"why not use directly the labelling model for the final classification, and instead we are using the results of the labelling model for obtaining a corpus to train from scratch new classifiers?"*. In fact, the Snorkel model is a classifier that outputs probabilities for each instance. One reason for not using Snorkel directly as our classifier is that the LFs are overfitted for the given dataset. Hence, the hypothesis is that the Snorkel model itself cannot generalize as well as a discriminative classifier trained on the probabilistic labels. Another problem is that we do not generate labels for each instance when all LFs abstain. We are just labelling a subset of the original data, the one better covered by the LFs. Thus, we need a model that makes discriminative decisions. Another point of view is that we are not using all the information in the text, just the one we decide to use in the LFs or the one for which we have domain knowledge LFs. A deep learning model can extract more features that will help them to learn why a text belongs to a category. Figure 4.2 shows a visual example of how a discriminative model can generalize from limited training data to being able to classify regions of the space not covered by the LFs.

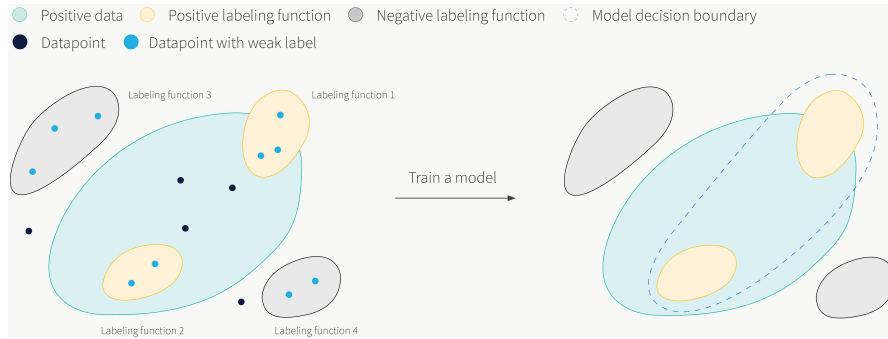


Figure 4.2: Diagram showing how a discriminative model can take advantage of limited training data generated with weak supervision. Source [17].

Furthermore, classification systems usually classify input texts (or whatever format it handles) individually independently to other inputs, and this labelling models may have LFs which depend on all the corpus (e.g. clustering methods) or which depends on the output of the LFs for other texts to determine the probabilistic results in the generative model. Not to mention the computational and time cost for each register that some LFs could need and that may be unacceptable for the requisites of many classification systems.

#### 4.1.1 Label functions

One of the key points on the design of a Snorkel architecture is defining LFs. Each LF can either assign a label or abstain for each sample. Also, a LF does not have to be prepared for assigning all the labels the problem has, it can only explore some characteristics and therefore have an output range of just one label or just some of them. Most common types of LFs are based on:

- Heuristics that check out different aspects of the samples, for example:
  - If it contains any or various determined keywords, expressions, symbols, etc.
  - Characteristics of the text like length, structure, length of sentences, appearance of elements like URLs, etc.
  - If it contains determined types of entities in the text (like names of people, places, organizations, etc).
- Clustering methods. The idea is to run a clustering algorithm to group the samples, and then select a criteria to assign labels to each cluster.
- Classification models. They take profit from existing models for similar problems, for example a model that predicts some of your goal categories but not for all, or apply with them transfer learning strategies.
- External knowledge. These LFs make use of external data to apply it for the decisions. For example check if the name of a person is in a defined list or check if some terms are contained in a certain database.

Snorkel offers certain metrics about the results of the LFs, like their coverage (the fraction of the samples that are labelled, or in other words, those samples in which they do not abstain), their overlapping (fraction of samples that a LF labels and other LF also labels) and conflicts (fraction of samples where the label assigned by the LF contradicts another LF's label). If gold labels are provided (samples with labels at the input that you already know are correct), we can also have metrics about the performance over them.

### 4.2 Design and implementation of the solution

#### 4.2.1 Definition of the input format

In order to facilitate working with the data, the first step is changing the format of the original dataset: each news item is given an unique ID and the news are grouped in JSON files with the following format showed in the figure 4.3.

```
1  [
2    {
3      "ID": 76, #This ID is unique among all files
4      "Title": "Más de 40 muertos en una jornada de
5              violencia en Pakistán",
6      "Description": "Una nueva jornada de violencia en
7                    Pakistán ha dejado más de 40 muertos, [...]"
8    },
9    { "ID": 77,
10      [...]
11    },
12    [...]
13  ]
```

Figure 4.3: JSON format of the input dataset.

Furthermore, during this transformation some news are filtered. There are several repeated news and a small percentage of news in the original datasets are in other languages besides Spanish, in special Galician and Catalan. Using the `langdetect` tool<sup>1</sup> these news are detected and not included in the transformed dataset. Also there are some faulty bodys that are empty or only contain the name of the journalist or the newspaper, so the news with a body shorter than 200 characters are also not included. So finally, the dataset is composed of several JSON files, each one with 50.000 news, up to a total of 438.029 news.

#### 4.2.2 Introduction to useful concepts and tools

First of all, this section makes an introduction about some of the concepts and strategies related to NLP expected to be applied to the weak supervision techniques to understand them and understand how they can be applied to design LFs.

- Text preprocessing is a common procedure applied before a lot of NLP tasks for facilitating the understating of the text. Some of the common preprocessing tasks are lowercasing the text, removing symbols (like @, #, (, ), -, etc.), removing stopwords (very common words such as articles, prepositions or conjunctions that

---

<sup>1</sup><https://github.com/fedeloopez77/langdetect>



are not distinctive elements and do not provide useful information), tokenizing the words (splitting the text into smaller units such as words or subwords), stemming (truncate the words to their root, e.g. 'troubling' to 'troubl') and lemmatization (transform the words to their base form, e.g. 'troubling' to 'trouble').

- Named entity recognition (NER) [18] is the task of identifying named entities from a text. A named entity (or just entity) is defined as a word or group of words that refer to any real-world object, either physical or abstract. NER systems find the entities in texts and classify them in categories such as locations, persons, organizations, date or time (e.g. "07/11/2020/" or "10:33 pm").
- Transformer [19] is an architecture for neural networks developed and popularized during the last years with plenty of applications in NLP. There are several available pre-trained Transformers models, standing out BERT [20], an state-of-the-art model for NLP systems.

The principal application of Transformers models is predicting the word that best fits in a given masked space in a text. It gives a set of words with their correspondent probability of fit there. These models are pre-trained with a huge amount of texts, making them learn for the task of, for every word, mask them and show the solution of the correct word to predict.

The pre-training step gives the model a high general knowledge of the language, that can be beneficial to adapt the model for performing different tasks or to adapt to specific datasets by a transfer learning process. This is done by a fine-tuning process, where more training data are involved, and can switch from the masked-word prediction task to other tasks such as text classification, text summarization, text generation, question answering, etc.

These are very heavy models, which need large time and powerful hardware resources for the pre-training steps, and therefore they are usually created only by big companies (Google, Microsoft, OpenAI, etc.). Their use for downstream tasks is feasible generally for standard users or companies, but it is clearly slower and computational more expensive than classic machine learning algorithms.

- TF-IDF (Term Frequency - Inverse Document Frequency) [21] is a representation in a numerical form of the relevance of a word in a document that is part of a corpus. Obviously the more times a word appears in the text, the more relevant is. But this is a measure referenced to a corpus, so also the more common in the corpus is the word, the less relevance it has. It has several applications, for example for information retrieval (finding relevant documents for a search, like search engines such as Google or Yahoo do) or text classification (classifying inputs based on the relevance of their words to the different categories).
- Latent Dirichlet Allocation (LDA) [22] is a topic modeling method for explaining the different categories or topics that can be associated with the documents of

a corpus. It is based on the premise that each document can be explained as a mixture of topics. A topic can be represented as a set of probabilities for each word, so the words with higher probability for a topic are the most representative of that topic.

As a first step, from the documents in the corpus it calculates multiple topics and then it calculates the probability of each document to belong to each topic based on the words of the document, considering a bag of words model (it means, the order does not affect). So, every document is represented as a mixture of various topics. From this point, it can be used as a clustering algorithm, considering that documents that belong to the same topics can be grouped or clustered together.

- Density based spatial clustering of applications with noise (DBSCAN) [23] is a common data clustering algorithm based on the position of a set of points in multidimensional space. It groups together points which are close to each other based on a distance measurement. This measurement is usually the Euclidean distance. For text corpus, each document can be represented as a point in a vectorial system, so the number of appearances of each word represents one dimension in the space.

It creates the clusters based on two main parameters: a minimum distance to other points for being considered neighbours (usually called epsilon) and on having a minimum number of neighbours for being considered part of a cluster. Unlike other clustering algorithms, in DBSCAN isolated points can be just considered as not being part of any cluster, and the resulting number of clusters is not configured.

Weakness of this method is working with sets of points in which different parts of the space have different densities of points, and also it usually struggles with high dimensionality data.

- Ordering points to identify the clustering structure (OPTICS) [24] is a clustering algorithm whose bases are similar to DBSCAN but it tries to solve the problem DBSCAN has with data of varying density by finding core samples of high density and expanding clusters from them.

There are multiple tools that facilitate and implement the application of techniques such as NER, text preprocessing and other NLP tasks. Some of the most common are Spacy<sup>2</sup>, NLTK<sup>3</sup>, Linguakit [25] and Stanza [26].

### 4.2.3 Proposal of label functions

The first step for the Snorkel implementation is providing a set of LFs. This section exposes the different ideas and implementations of LFs, their main characteristics and

---

<sup>2</sup><https://spacy.io/>

<sup>3</sup><https://www.nltk.org/>

their fitness for this classification problem. Some of these LFs are discarded after analyzing the performance in preliminary tests. We motivate this decision in this section too.

For each LF there will be some comments about their expected performance. For example if they are expected to have a big or small coverage, if they prioritize one metric over the others, or if they are expected to expend a big or small execution time. These comments are just general impressions and the specific values for the metrics will be evaluated later in the results sections.

A small set of labelled news is used in some of the LFs as a training set or for helping to recognize different characteristics of the categories. This set has no relation with the dataset we use in this problem. It has been made up by the collection of labelled texts from different digital publications as newspapers, blogs and magazines, selecting them from the specific sections of the publications or from publications that only address themes of a specific subject (sports press, science magazines, etc). This external dataset is different and much smaller than the one we want to categorize, having a different distribution and just few thousands of news, so there are just some hundreds of news of each category.

One of the characteristics analyzed in these labelled news are the most common words or bigrams in each category for discovering keywords that can later be used for creating LFs.

### 4.2.3.1 Heuristics based on keywords

The first LFs proposed are a set of LFs based on heuristics that use keywords. There is one LF for each category.

The LF uses a simple heuristic that checks a list of keywords and if any of them is in the text assigns the corresponding label to the news item and if none of them matches, it abstains. An example of this is represented in the figure [4.4](#).

There are just a few keywords for each category (about 10-15) in order to optimize the precision of the labeller and to try to reduce the mistakes. Therefore, the keywords are very common words in each category and if any of them could produce confusion, it is avoided (e.g., the word *partido* is very common for *sports*, but also for *politics*, so it is avoided in both).

These LFs are expected to have a small coverage (they only aim to label for one category) but the precision is expected to be high. Also the execution of this heuristic should be fast because it just needs to check appearances in the text.

Each LFs will be designated an identifying name for being named during this memory. In this case, the names of the LFs have the structure LF\_keywords\_<label>, for example LF\_keywords\_sports or LF\_keywords\_science.

Keywords of each category			
<b>POLITICS</b>	Diputados, ministro, elecciones, congreso	✓ Match in the category	✗ Abstention
<b>SCIENCE</b>	Ciencia, científicos, biología, medicina		

Text	<b>POLITICS</b>	<b>SCIENCE</b>
Los <b>diputados</b> se reúnen para debatir sobre la nueva Ley de Eutanasia	✓	✗
Los <b>científicos</b> avalan la eficacia de las vacunas	✗	✓
El <b>ministro</b> de <b>ciencia</b> comparecerá ante los medios el próximo martes	✓	✓

Figure 4.4: Diagram with the keywords based LF.

### 4.2.3.2 Heuristic based on sports professionals names

This LF is specific to detect sports news. It checks a list of names of professional men and women from the field of sports, and if any of those names are contained in the text it labels it as *sports* and otherwise it abstains, as shown in figure 4.5. It also contains some names of clubs or teams. In total the list has a size of 1456 names.

The list contains names from:

- List of players in Spanish men's football league (1st and 2nd division) in season 2020/2021. Also names of clubs (excluding the names that are exactly equal to the names of cities, to avoid confusion with news about non sports events in those cities, e.g. Valencia or Málaga).
- List of best players in NBA in 2017.
- List of top 100 tennis players of both ATP (Association of Tennis Professionals) and WTA (Women's Tennis Association) rankings.
- List of the best paid sportspeople in the world in 2010 and 2014 according to Forbes.
- List of Spanish medalists in the Olympics Games in the XXI century.

As the previous one, this LF should be very fast and should have a small number of false positives, because the vast majority of news that mention sportspeople are about sports. But due to the limitation of the list, and the fact that not all sports news mention the names of famous people, there will be probably a lot of sports news not caught by this LF. So, the expected metrics are high precision and moderate recall.

The name for this LF is LF\_names\_sports.

## 4.2. Design and implementation of the solution

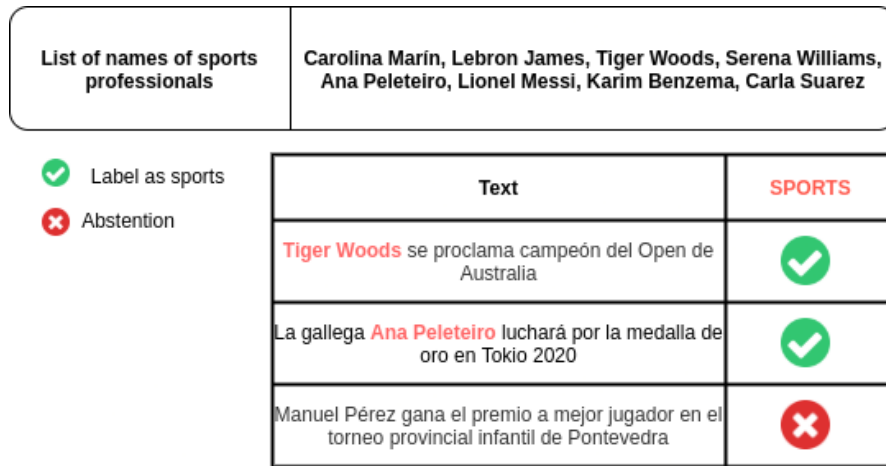


Figure 4.5: Diagram with the LF based on sports professionals names.

### 4.2.3.3 Heuristic based on references to monetary quantities

These LFs checks if in the text are references to big monetary quantities, frequent in economical topics, such as *millones de dólares*, *millones de euros*, *X euros*, *X libras*, etc., being X a number. In case it matches any of the patterns, it labels as *economy*, otherwise it abstains, as shown in figure 4.6. Again, this LF should be fast, and regarding to the

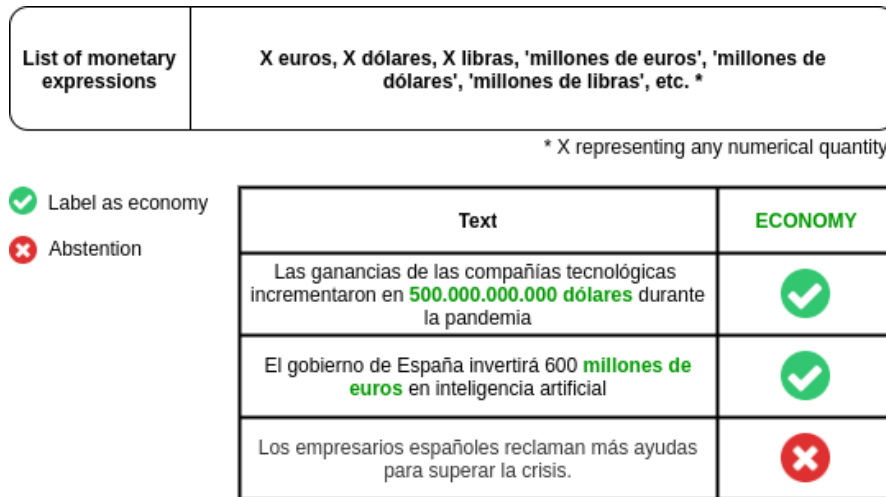


Figure 4.6: Diagram with the LF based on monetarian expressions.

metrics there is not any reliable supposition, but hopefully it will perform well. The validation process will confirm if this happens.

The name for this LF is LF\_monetarian.

### 4.2.3.4 Logistic regression classifiers with TF-IDF representations

This is a set of LFs consisting on a classifier for each category. It is based on a logistic regression that uses a TF-IDF representation of the texts, and each of them discriminates the corresponding category against all the others (e.g. *sports* vs *not\_sports*) and the LF will abstain when it identifies news in the group of ‘others’.

The input of the classifier consists, for each news item, of a vector with the TF-IDF representation of the text. Before applying the transformation to vectorize the texts, it is applied over them some preprocessing, based on deleting the stopwords and tokenizing the text. This pipeline is shown in figure 4.7.

The classifier is a logistic regression model. It is trained with texts from the external dataset mentioned at the beginning of this section. For each classifier, the training set consists of the text of that category (X), and a random subset of the texts of the rest of categories, these last ones labelled as ‘NOT\_X’. The number of texts of the ‘NOT\_X’ category in the train sets has been optimized with a validation process called cross-validation<sup>4</sup>. In table 4.1 we can see the metrics for each category for this cross validation process with different balance of labels, where for example the 2.5 column means that there are 2.5 ‘NOT\_X’ news in the train set for each news item of the correspondent ‘X’ label.

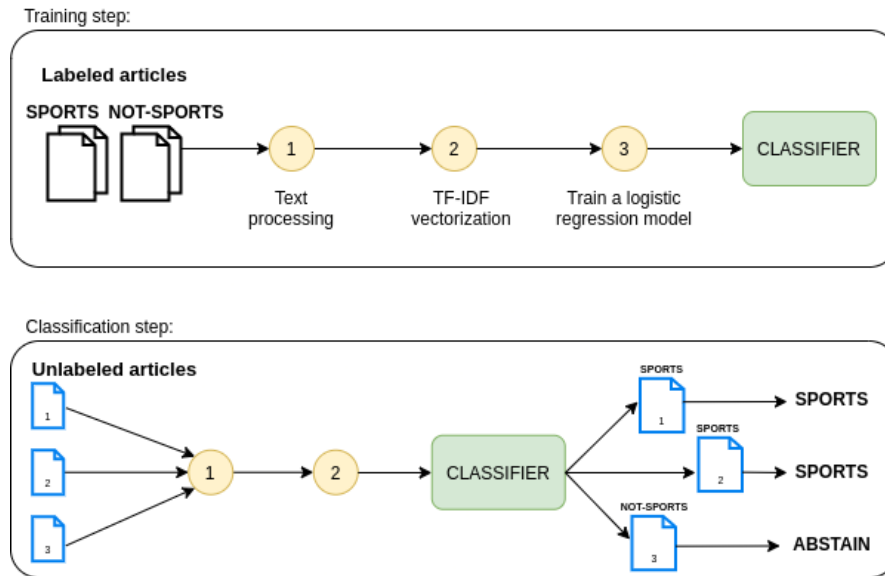


Figure 4.7: Diagram with the workflow of the TF-IDF, logistic regression based LF.

From these results of the cross-validation we can find in all the tested scenarios a high precision, but lower recall in general for all labels for the more unbalanced cases (2.5 and 1.8). For the more balanced ones (1.2 and 1), the precision is a bit smaller but recall

<sup>4</sup>[https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)

## 4.2. Design and implementation of the solution

increases significantly, and therefore so does the F1-score. So between these two last scenarios the F1-score has just small differences, but in the interest of optimizing the precision instead of the recall, there will be selected a balance of 1.2 for the training sets of all the categories.

So, as conclusion, this is a set of LFs that are expected to have high precision, and also being fast to calculate labels. Although, it can be expected that it will achieve different results testing with different test sets compared to the cross-validation results, but this will be checked afterwards.

The names for these LFs have the structure LF\_tfidf\_<label>, such as LF\_tfidf\_politics or LF\_tfidf\_sports.

	2.5			1.8			1.2			1		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Economy	92	38	53	89	50	64	89	77	82	81	81	81
People	97	25	40	98	39	56	98	55	70	98	59	74
Politics	100	47	64	100	62	76	96	79	86	87	87	87
Reports	96	71	82	96	75	84	86	87	86	81	91	86
Science	92	27	41	95	40	56	88	62	73	81	67	73
Sports	96	79	87	95	83	89	95	86	90	90	91	91

Table 4.1: Cross validation metrics for different ratio of labels in the TF-IDF based LF. Precision (P), Recall (R) and F1-score (F1)

### 4.2.3.5 Prediction of masked words with Transformers

Transformers can be used, as explained before, to predict the word that best suits a masked space in a text. It can be used for this problem in different ways. The proposal for this LF is give to the model the text to label, preceded or followed by a prompt, such as “This news is from the section <mask>.”, so it will give you the words with higher probabilities to fit in the masked space according to its prediction, and a label will be assigned if these words match with some defined keywords related to each label. Figure 4.8 shows a diagram of how this LF works.

After testing with some different prompts, we saw that there are some very recurrent predictions that fit in the categories, so the list of keywords to check was constructed looking to those results. There are also a lot of news where the model gives predictions that do not make sense with the expected results, so in those cases as they do not match any keyword, the LF abstain. Also sometimes the predictions makes sense but it do not give useful information for this problem, for example predicting a word related to the geographical location where the event in the news happens, so in these cases the LF also abstains.

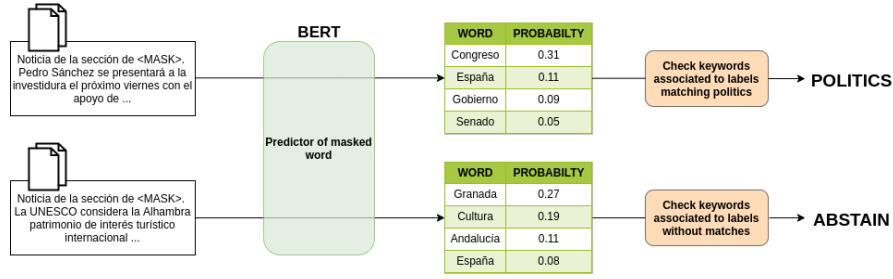


Figure 4.8: Diagram with the workflow of the BERT based LF.

Any of the Transformers models<sup>5</sup> for Spanish language could be used for this purpose. Here bert-base-multilingual-cased<sup>6</sup> and distilbert-base-multilingual-cased<sup>7</sup> [27] have been considered, using finally the BERT model. The table 4.2 shows different prompts tested and the percentage of abstention they reached using the selected keywords for the 10.000 first news of the dataset. Finally just the number 1 and 3 have been selected for a LF each of them. The number 2 has been dismissed due to its high abstention rate.

Regarding the duration, this is an expensive LF, it takes a lot of time to analyze and predict for each news. Also the expectation is to achieve a medium coverage, similar to the one in the preliminary tests (table 4.2).

So, for the two LFs, one for each prompt, the names for these LFs are LF\_BERT\_1 and LF\_BERT\_3.

	Prompt	Abstention
1	"Sección de <mask>." + text	30
2	"Noticia sobre <mask>." + text	87
3	"Noticia de la sección de <mask>." + text	32

Table 4.2: Table with the different prompts used to test BERT based LF.

<sup>5</sup><https://huggingface.co/models>

<sup>6</sup><https://huggingface.co/bert-base-multilingual-cased>

<sup>7</sup><https://huggingface.co/distilbert-base-multilingual-cased>



### 4.2.3.6 Classifier based on the entities in the news.

This LF consists of an algorithm that works with the concept on entities. It is divided in two steps:

1. The first step consists of identify the entities in the texts, which is done with the Spacy tool for entity recognizing using the model *es\_core\_news\_md*<sup>8</sup>. Also defining a set of keywords for each category. Then, for each entity in the texts, it counts how many times each of the keywords appears in the same text as the entity, so then it assigns a category to each entity based on the category of the most repeated keywords that appear next to it.
2. The second step is classifying the texts. Just find the entities in the text and calculate the most repeated category associated to those entities.

Figure 4.9 shows a simplified example of how it would work.

Ideally the training phase should be done with all the text of the dataset, but in terms of reducing its computing time it can be done just with a sufficient representative subset.

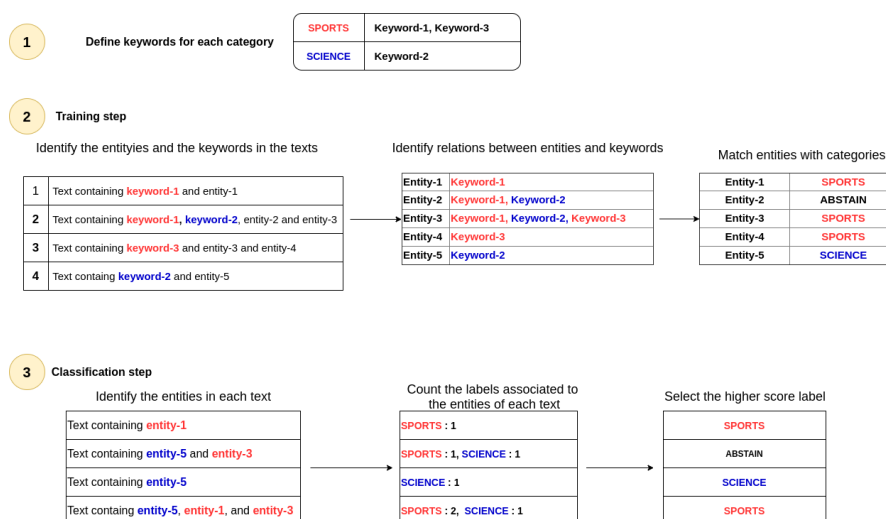


Figure 4.9: Diagram with the workflow of the entity based LF.

This will be a time expensive LF because the entity extraction for that many texts takes its time. One option to improve performance is storing the entities of each text in the training phase so that they are already known for the second phase, and also for other LFs that may eventually need to extract them.

The identifying name for this LF is LF\_Entities.

<sup>8</sup><https://spacy.io/models/es>

### 4.2.3.7 Classifier based on the most repeated professions by category

This LF is based on a list of hundreds of names of professions. First of all, it compares the number of apparitions of each profession in the labelled set of news that was already used in other LFs, giving a weight for each category for each profession. Then during classification time it just looks for the names of professions contained in the corresponding text and calculates the category with a higher score due to the weights. This process is represented in the diagram in figure 4.10.

This LF should be quick, not as instantaneous as the keyword based ones, but should not take too much time.

The name for this LF is LF\_professions.

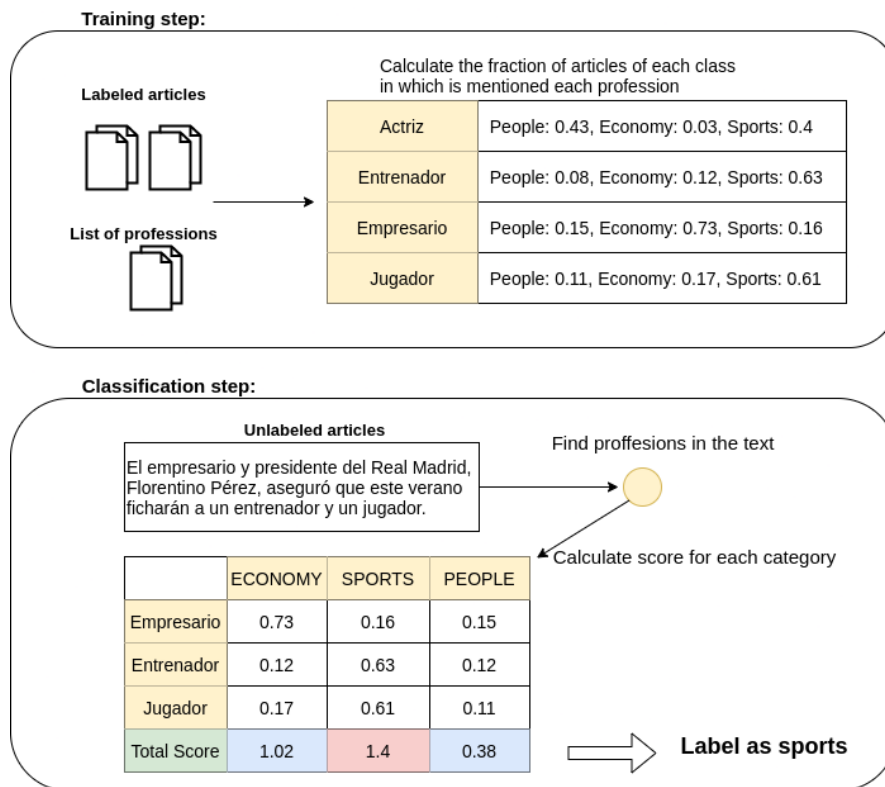


Figure 4.10: Diagram with the LF based in professions names.

### 4.2.3.8 LDA clustering based LF

Here is introduced the use of clustering methods for group the texts. The main idea is to apply a clustering method to ideally place the news in groups where all of them are of the same category or talk about similar topics, and then analyze those clusters and find the way to label them.

This LF is based in LDA. The workflow of the LF (showed in figure 4.11) starts with the text vectorization of the documents of the corpus, for applying the LDA method over them. From the set of texts, LDA first creates a number of topics, and then for each document calculates the words that best represent its content.

Then each topic is assigned to a category from the taxonomy (or abstain) based on the most common words of the topic. An intuitive solution is to base this assignment on keywords of each category.

Finally it labels each document, combining the labels of the top 15 most representative topics for it, and the probabilities of pertaining to each of them, and assigns the label with a higher probability.

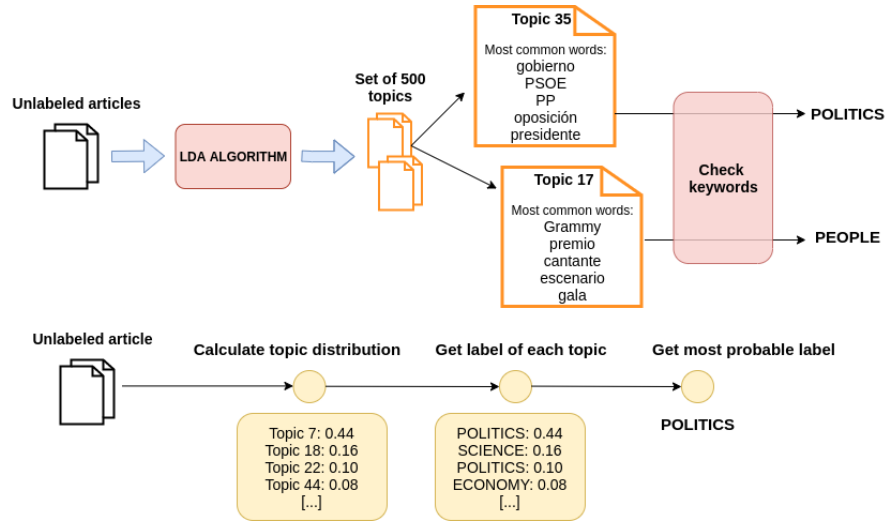


Figure 4.11: Diagram with the workflow of the LDA based LF.

The LDA algorithm has been executed in various scenarios, varying the number of topics to find and changing the size of the input dataset.

One of the conclusions obtained is that clearly the topic creation improves when the size of the dataset increases (something that can be expected), obtaining topics where the most relevant words show very specific subjects, which is desirable in terms of matching them with the labels. In a first attempt with just 10.000 news, these topics were sometimes a bit abstract, mixing among their more important words terms that are not about the same

subjects. After enlarging the dataset size to 60.000 news, the most representative words of each topic are in the majority of cases very related between all of them, showing that the topic expresses a concrete theme. This improvement is even bigger when enlarging to 150.000 or more news.

Regarding the selection of the number of topics, given the big size of the input dataset and the goal of having topics as much specific as possible, the best option is to configure the algorithm to return a big number of topics, even though the number of categories to label is small. After different attempts, a size of 500 topics have been selected. It leads to a lot of topics with few texts associated because they address unusual themes, and other topics about very common subjects with a lot of news associated.

One important aspect about the implementation is the tool to use. The LDA implementation of the library Scikit-learn<sup>9</sup> can give a lot of problems when scaling to several thousands of texts and hundreds of topics, indeed, in tests during the implementation the program crashed not permitting to finish the execution with a set of 10.000 texts. Otherwise, the implementation of the Gensim library<sup>10</sup> scaled well, calculating the topics in an acceptable time. The duration of the execution of the LDA process took about 2h for the whole dataset. For cases with bigger datasets, one option could be just try to run the LDA implementation with all the texts, or run it for a smaller but big enough representative subset and then just “classify” the texts: calculate the topic distribution as it is done with the ones used for that topic calculation.

The name for this LF is LF\_LDA.

### 4.2.3.9 DBSCAN and OPTICS

Despite of the potential problems it usually has with high dimensional data (like the data in this problem), there have been explored ways to use DBSCAN for an LF to prove if it can be useful or not. Scikit-learn implementation was used, and the algorithm was executed with multiple combinations of the input parameters, finding always some of the following problems: either it ‘abstained’ for most of the samples, or it considered just one really big cluster ( $> 50\%$  samples) and isolating the rest of the samples, or considered a lot of very small clusters (just few texts) which is not useful. This is reasonable because of the probably wide spectrum of density values of ‘points’ (texts) and the big amount of dimensions (each word is a dimension). So due to those reasons, this potential LF has been dismissed.

Also there has been an attempt to design an LF that uses OPTICS that perhaps could solve the problems found by DBSCAN. Even though, in this case the results achieved were very similar to the previous DBSCAN case, so the use of this algorithm was also

---

<sup>9</sup><https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html>

<sup>10</sup>[https://radimrehurek.com/gensim/auto\\_examples/tutorials/run\\_lda.html](https://radimrehurek.com/gensim/auto_examples/tutorials/run_lda.html)

dismissed.

## 4.3 Snorkel framework implementation

The LFs have been developed as independent weak classifiers from the Snorkel framework, in order to simplify the execution and facilitate running it several times with small variations. For example run the framework again removing some of LFs to compare performances.

So, first of all, the LFs are executed and as a result they give files which map the ID of each news item with the predicted label by the corresponding LF. Then during the Snorkel workflow the task for each LF is just to read those files and adquire the labels for each ID. So the workflow remains as the figure 4.12 describes.

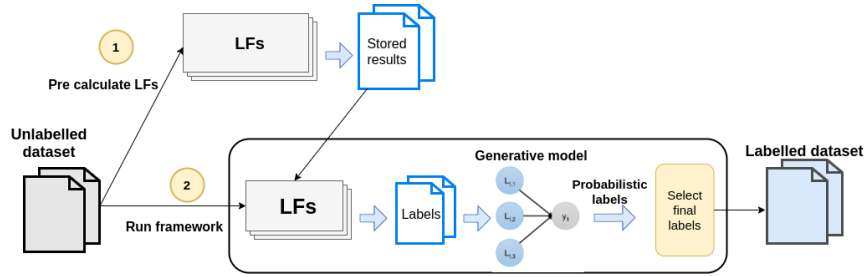


Figure 4.12: Diagram with the implemented workflow for Snorkel.

Once the framework reads the labels from all the LFs, it has to aggregate them to obtain the final results. For this, we will run the framework few times following different strategies:

- First of all, following a majority vote strategy. This means that the selected final label for each news item is the one most repeated among the labels generated by all the LFs. It will be calculated to use as a baseline to compare the results with the ones of the generative model, expecting these last ones to be better. Also, this strategy is not really fair in scenarios like this, where there are a different number of LFs that can assign each label, giving more chances to some labels than to the others. This strategy will only select abstention when, either there is a tie or all the LFs have abstained.
- Using the Snorkel generative model for obtaining a probability for each label. In some scenarios it could be enough to take the most probable label for each news item, but for our case, where we know that some of the news do not belong to any of the categories, we need to discern when and why consider a news item as abstention for not considering it into the final training set. For this decision, we will make an statistical analysis of the probabilities given in further sections.

- Also it is interesting to make analysis and comparisons against the results of executing the framework not using some of the LFs. Specially without the BERT based ones, that are, a priori, the less time and resource efficient, to see if it is worthy to expend the time running them or not. Also, based on the results and metrics of the LFs, it could arise the interest in make a similar comparison related to some other LF. This is called an ablation study.

Finally, we can use the test set to validate the final results.

## 5 Results and evaluation

### 5.1 Evaluation of label functions

#### 5.1.1 Results of individual label functions

First of all, we will show the results achieved by the LFs in the test set. For each LF it is displayed a table with its precision, recall, f1-score and accuracy in the test set and, excepting for the LFs that only have one possible category to label, it is displayed the confusion matrix, that shows the numerical correspondence between the predicted labels and the real labels in the test set.

The important point of each LF is not achieving awesome values in the metrics, but showing a positive tendency. We can identify the positive tendencies in a confusion matrix when the diagonal tends to be highlighted. Also the boundary for a LF for being useful is performing better than a random label generator [28]. For this case, the table 5.1 shows the metrics for this problem's test set of a random classifier that labels randomly following the distribution of labels in the test set. The metrics consist of the average of 500 executions of this random classifier. We can see they are very low values, so we expect our LFs to outperform the random baselines by far.

##### 5.1.1.1 Keywords LFs

Table 5.2 shows the metrics for the specific labels for which are designed each LF. For the LFs based on keywords we can see a very variant precision (0.33 - 0.71) depending on the category and in general a recall a bit higher (0.54 - 0.83). Clearly these LFs are not great by themselves, but the nature of Snorkel is having noisy labels so they will be completely useful.

The LF based on monetarian quantities shows very poor performance, labelling a small fraction of the *economy* news and also failing a lot in the ones that considers as being

	precision	recall	f1-score
ABSTAIN	0.096	0.096	0.096
ECONOMY	0.121	0.122	0.122
PEOPLE	0.070	0.070	0.070
POLITICS	0.311	0.312	0.311
REPORTS	0.247	0.246	0.247
SCIENCE	0.040	0.039	0.039
SPORTS	0.120	0.120	0.120
accuracy			0.201
macro avg	0.144	0.144	0.144
weighted avg	0.201	0.201	0.201

Table 5.1: Metrics of a random baseline classifier

about economy. According to the poor results, is reasonable to doubt about the utility of this LF, so it will be discussed later its influence and the possibility of discard it.

In the other hand, the LF based in recognising names of sportspeople has the expected results: a good precision because most of the news that contain one of the names in the list are about sports, and a low recall, because the list is limited. Although, this last fact fits in the expected behaviour, and this LF will help Snorkel to reinforce decisions for the *sports* category.

	Label	precision	recall	f1-score
LF_keywords_economy	ECONOMY	0.330	0.543	0.411
LF_keywords_people	PEOPLE	0.519	0.718	0.602
LF_keywords_politics	POLITICS	0.715	0.784	0.748
LF_keywords_reports	REPORTS	0.685	0.734	0.708
LF_keywords_science	SCIENCE	0.388	0.826	0.528
LF_keywords_sports	SPORTS	0.655	0.838	0.735
LF_keywords_monetarian	ECONOMY	0.378	0.243	0.296
LF_keywords_sports_names	SPORTS	0.793	0.338	0.474

Table 5.2: Metrics of the keywords LFs

#### 5.1.1.2 TF-IDF LFs

In the case of the logistic regression and TF-IDF based LF, the table 5.3 shows the metrics for the correspondent specific labels. Compared to the previous keywords LFs metrics, for all the categories we can see this works better according to the metrics. We can highlight the results for *sports* category where it achieves a 100% precision.



## 5.1. Evaluation of label functions

	Label	precision	recall	f1-score
LF_tfdif_economy	ECONOMY	0.864	0.543	0.667
LF_tfdif_politics	POLITICS	0.861	0.739	0.795
LF_tfdif_people	PEOPLE	0.742	0.590	0.657
LF_tfdif_reports	REPORTS	0.824	0.878	0.850
LF_tfdif_science	SCIENCE	0.739	0.739	0.739
LF_tfdif_sports	SPORTS	1.000	0.882	0.938

Table 5.3: Metrics of the TF-IDF LFs

### 5.1.1.3 LDA\_LF

The confusion matrix of the evaluation of this LF’s results (in figure 5.1) shows clearly a highlighted diagonal, which is a good signal at first sight. It means a significant fraction of news are correctly classified. The other highlighted part of the matrix is the column of news predicted as abstention. We can see the majority of misclassified news are considered as abstention, and this is better for our purpose than if they were misclassified with other label, because simply later the Snorkel model will not take them into account.

Regarding to the metrics (table 5.4), once again it works better for *sports* than for any other, and in this case the poorer results are found for *economy* and *people*. But as stated before, the important point in this LF is that the most of the misclassified news are not with another label, just abstentions.

	precision	recall	f1-score
ABSTAIN	0.149	0.611	0.240
ECONOMY	0.679	0.271	0.388
PEOPLE	0.467	0.359	0.406
POLITICS	0.816	0.528	0.641
REPORTS	0.911	0.590	0.716
SCIENCE	0.593	0.696	0.640
SPORTS	0.966	0.838	0.898
accuracy			0.552
macro avg	0.654	0.556	0.561
weighted avg	0.744	0.552	0.605

Table 5.4: Metrics of LF\_LDA

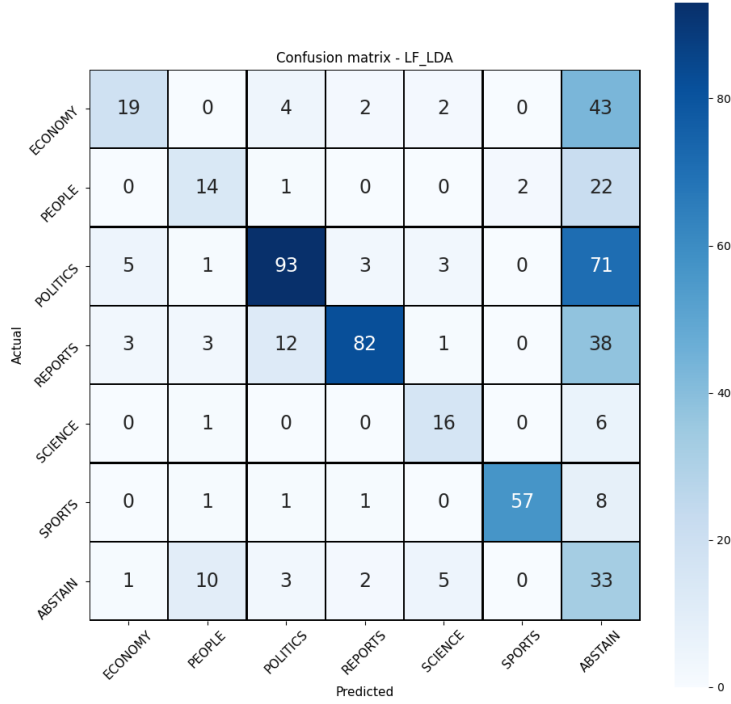


Figure 5.1: Confusion matrix for LF\_LDA

### 5.1.1.4 Entities LF

The first thing that should take our attention from the confusion matrix of the entities based LF in figure 5.2 is that it is not diagonal-shaped. The column of *politics* prediction is the one standing out, what means that the LF has a predisposition to label as *politics* most of the news. We can see that there are almost no abstention cases. We can see, in both the matrix and the metrics table (table 5.5), that *sports* category is the one that better resists the *politics* "invasion", meanwhile *economy* is fully eclipsed by *politics* and gets not even one news item labelled as *economy*. Even though this LF has not convincing results, it is better than the random baseline, so theoretically it has some utility. In any case, this will be evaluated later to realize the effect it can produce or not in the final results.

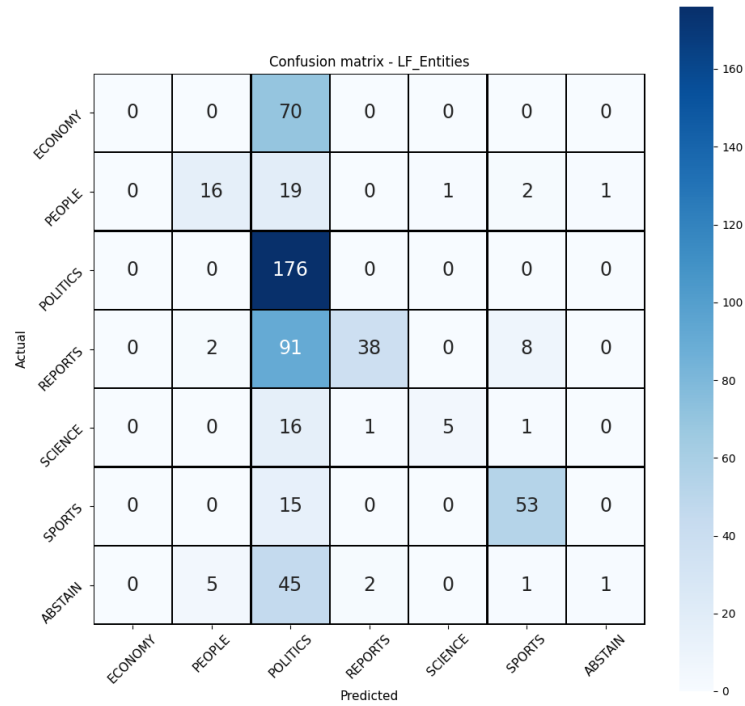


Figure 5.2: Confusion matrix for LF\_Entities

	precision	recall	f1-score
ABSTAIN	0.500	0.019	0.036
ECONOMY	0.000	0.000	0.000
PEOPLE	0.696	0.410	0.516
POLITICS	0.407	1.000	0.579
REPORTS	0.927	0.273	0.422
SCIENCE	0.833	0.217	0.345
SPORTS	0.815	0.779	0.797
accuracy			0.508
macro avg	0.597	0.386	0.385
weighted avg	0.579	0.508	0.430

Table 5.5: Metrics of LF\_Entities

### 5.1.1.5 Professions LF

Regarding the LF based on a classifier that takes into account the frequency of names of professions in the texts of each category, we see a confusion matrix in the figure 5.3 with not as many highlights as some of the previous ones. We can see that in most cases the biggest value in rows or columns are in the diagonal, and that there are not too many news misclassified as abstention. Again we can see a tendency to misclassify as *politics*, but clearly lighter than in the case of the previous LF\_Entities. Looking to the specific values for each label in the table 5.6, none of them have awesome metrics. The best metrics f1-score are for *sports* and *politics*, and *economy* and *science* have the worst results.

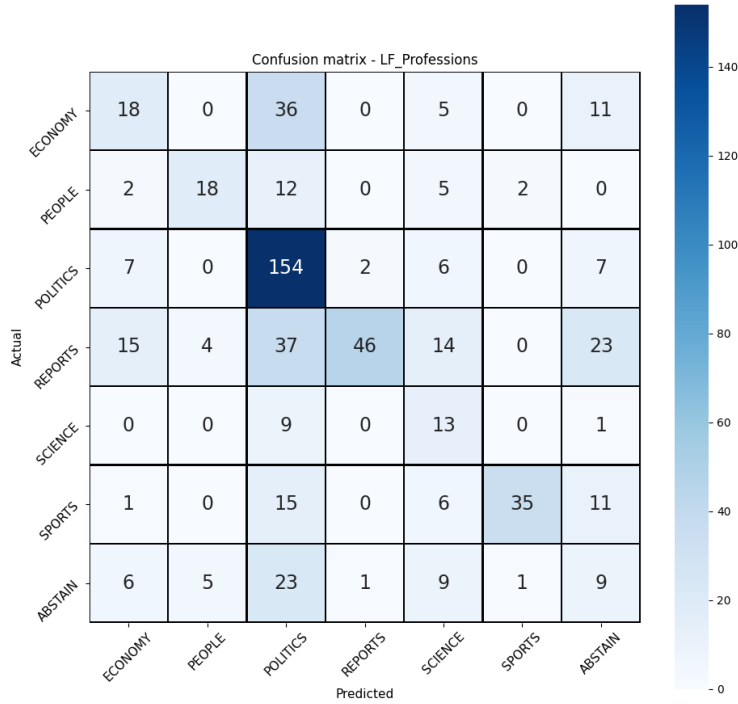


Figure 5.3: Confusion matrix for LF\_Professions

	precision	recall	f1-score
ABSTAIN	0.145	0.167	0.155
ECONOMY	0.367	0.257	0.303
PEOPLE	0.667	0.462	0.545
POLITICS	0.538	0.875	0.667
REPORTS	0.939	0.331	0.489
SCIENCE	0.224	0.565	0.321
SPORTS	0.921	0.515	0.660
accuracy			0.515
macro avg	0.543	0.453	0.449
weighted avg	0.620	0.515	0.507

Table 5.6: Metrics of LF\_Professions

#### 5.1.1.6 BERT LFs

In the confusion matrix of the LF using BERT with the prompt n<sup>o</sup> 1 ("Sección de <mask>"), in the figure 5.4, we can recognise a clear highlighted diagonal representing a lot of correctly classified news and also big values in the columns of news where the predictions abstain. In the table of figure 5.7 we can see the metrics for each label, where we can note a different range of values, where we can highlight the very high precision for the *reports* label that is close to 100%.

Regarding to the LF that uses the prompt n<sup>o</sup> 3 ("Noticia de la sección de <mask>"), we can see in figure 5.5 a similar confusion matrix, with the characteristic that this time there are a lot of news misclassified as *sports*. In the metrics table (table 5.8) we see the accuracy is worse than with the other prompt.

	precision	recall	f1-score
ABSTAIN	0.116	0.296	0.167
ECONOMY	0.603	0.586	0.594
PEOPLE	0.465	0.513	0.488
POLITICS	0.696	0.676	0.686
REPORTS	0.983	0.417	0.586
SCIENCE	0.444	0.522	0.480
SPORTS	0.810	0.750	0.779
accuracy			0.557
macro avg	0.588	0.537	0.540
weighted avg	0.687	0.557	0.590

Table 5.7: Metrics of LF\_BERT\_1

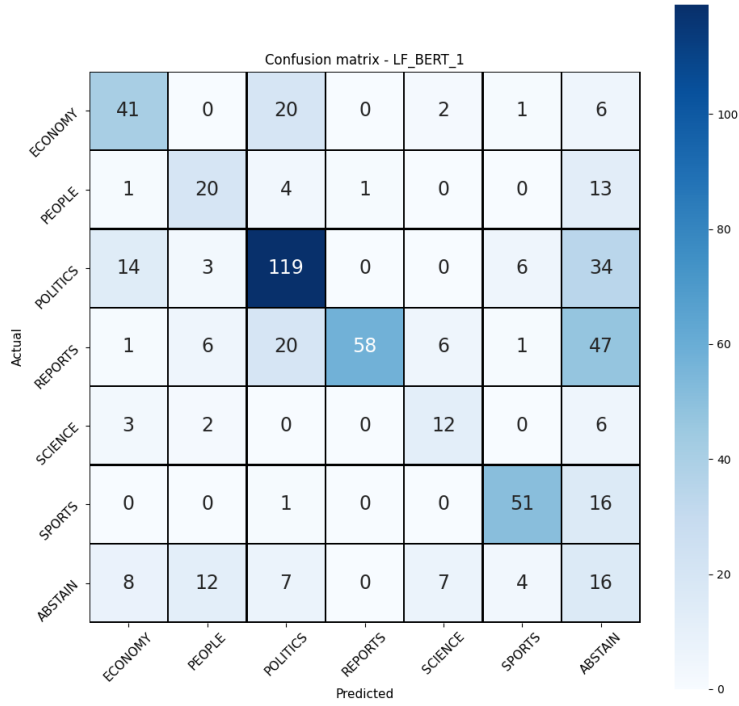


Figure 5.4: Confusion matrix for LF\_BERT\_1

	precision	recall	f1-score
ABSTAIN	0.110	0.204	0.143
ECONOMY	0.700	0.400	0.509
PEOPLE	0.463	0.641	0.538
POLITICS	0.607	0.483	0.538
REPORTS	1.000	0.424	0.596
SCIENCE	0.615	0.348	0.444
SPORTS	0.387	0.926	0.545
accuracy			0.490
macro avg	0.555	0.489	0.473
weighted avg	0.631	0.490	0.508

Table 5.8: Metrics of LF\_BERT\_3

## 5.1. Evaluation of label functions

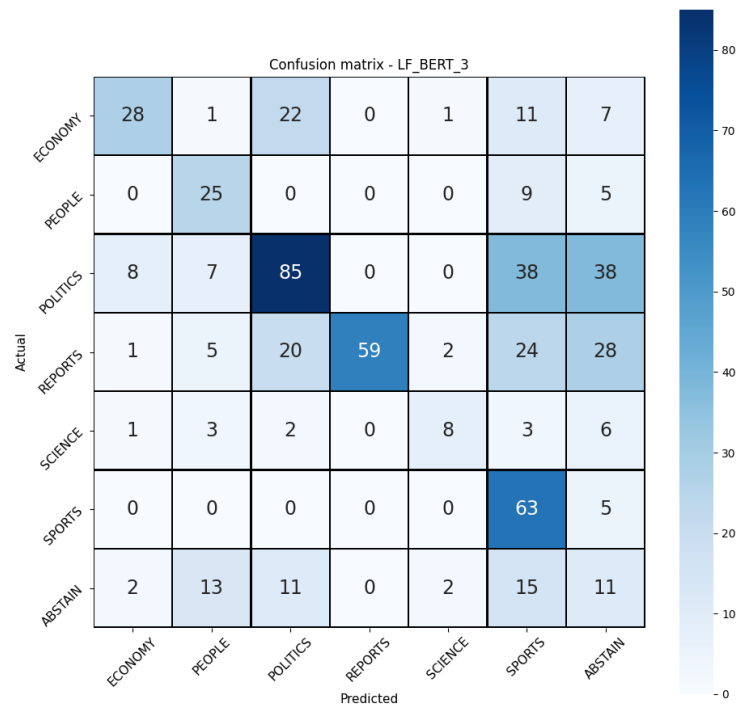


Figure 5.5: Confusion matrix for LF\_BERT\_3

### 5.1.2 Statistics of label functions in the full dataset

The table 5.9 shows some statistics and data about the execution and results of the LFs:

- First of all, the coverage is the percentage of the news of the dataset that have been labelled by the LF, or in other words, the percentage of news in which it do not abstain. As we can see, the values are very diverse. The LFs that only target one category have smaller values, and among them they approximately follow the distribution of the dataset. The coverage of the LFs for *politics* or *reports* are bigger than the ones for *science*, for example.
- The conflicts column shows the percentage of conflicts in the labels of the LF for the whole dataset. A conflict means that any of the other LFs has labelled the same news item into another category (not counting abstention). Obviously this value can not be bigger than the coverage, and we see that in most of the cases is smaller but close to the coverage value. It indicates that in most of the news there are disagreements between the LFs. In further sections we will study more about these conflicts.
- Duration column shows the aproximate average time it took for calculating the labels for 50.000 news. These measures are relative to a Ubuntu 18 machine with 8 RAM GB, an Intel i3 330GHz processor and no GPU. In the LFs that have a training or precalculation step, the training time refers to the total time expent in that process, and only the prediction time is relative to size of 50.000 news. The most time expensive LFs are the ones based in BERT, which for the dataset of more than 400.000 news would take several days to run. So in order to speed up, these LFs have been runned in a high performance virtual machine located in a server from CESGA<sup>1</sup> (Centro de Supercomputación de Galicia), where it calculated at a 315% higher speed, taking about 3.5 hours to calculate 50.000 predictions. Despite of this, the value included in the table is the assumption of running it in the same machine as the rest of LFs, for making a fair comparison.

Adding up all the values of duration of the execution of the LFs, we can find the total time expent in obtaining the LF labels. Considering the time of the BERT LFs the one took in the CESGA machines, the total time would be of apporximately 91 hours (3 days and 19h). Without taking using the BERT LFs it would have taken just about 30h (1 day and 6h), so an interesting analysis that we will make is to compare the results of Snorkel using and not using those LFs to check if the time they need is worth it or not.

Finally, heatmap in figure 5.6 shows which labels had more conflicts among them in the predictions of the LFs. Compared with figure 3.8 that showed the conflicts in the predictions made by humans, we can identify some correlations. In both cases the

---

<sup>1</sup><https://www.cesga.es/>



### 5.1. Evaluation of label functions

	% Coverage	% Conflicts	Duration (50.000 news)
LF_keywords_economy	19.7	19.7	2.5 min
LF_keywords_people	9.7	9.0	2.5 min
LF_keywords_politics	30.4	23.5	2.5 min
LF_keywords_reports	20.9	19.4	2.5 min
LF_keywords_science	9.4	9.1	2.5 min
LF_keywords_sports	12.1	9.5	2.5 min
LF_keywords_monetarian	11.6	11.6	2.5 min
LF_keywords_sports_names	4.3	2.8	2.5 min
LF_tfidf_economy	8.8	8.8	Train: 2 min, Predict: 2 min
LF_tfidf_people	3.0	2.4	Train: 2 min, Predict: 2 min
LF_tfidf_politics	24.6	17.6	Train: 2 min, Predict: 2 min
LF_tfidf_reports	23.7	21.9	Train: 2 min, Predict: 2 min
LF_tfidf_science	5.0	4.7	Train: 2 min, Predict: 2 min
LF_tfidf_sports	7.8	4.7	Train: 2 min, Predict: 2 min
LF_LDA	55.1	45.4	Train: 2h, Predict: 8 min
LF_Entities	99.2	82.6	Train: 5h, Predict: 1h
LF_professions	88.1	74.2	Train: 2h, Predict: 45 min
LF_BERT_1	77.6	65.5	11h
LF_BERT_3	81.7	69.3	11h

Table 5.9: Statistics of the label functions in full dataset

most conflicting pair is *politics* and *economy*, showing that for both humans and for the classifiers is sometimes difficult to distinguish between news of those categories. The other significant characteristic is that *politics* conflicts a lot with all the other categories. It makes sense because some of the LFs that we have seen tend to misclassify news of other categories as *politics*.

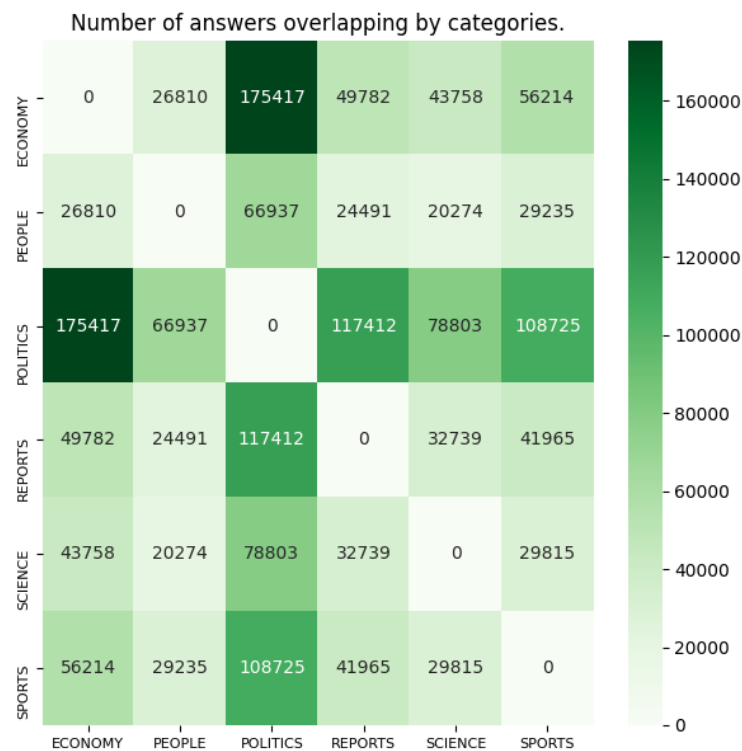


Figure 5.6: Conflicts among labels in the LFs outputs.

## 5.2 Evaluation of aggregation of label functions

### 5.2.1 Majority vote model

For the analysis of the results of the execution of the framework, first of all we are going to check the majority vote strategy. We see in figure 5.7 multiple histograms. Each of them corresponds to the news where the most voted category by the LFs is the indicated, and represents the number of news corresponding to each fraction of votes labelling in that category. Nevertheless, in figure 5.8 we can see a confusion matrix with the diagonal highlighted and few errors, excepting the related with abstentions, and some news misclassified labelled as *politics* but actually belonging to *economy* and *reports* categories, which again, as commented previously and showed in figure 3.8, are expectable due to the nature of the dataset and the overlapping of categories in some news.

Regarding to the metrics (table 5.10), this achieves a total accuracy of 0.735, showing better results of f1-score in categories such as *sports* (0.948) and lower in *economy* (0.642) and *people* (0.675). These results will be used as a baseline to compare with other strategies.

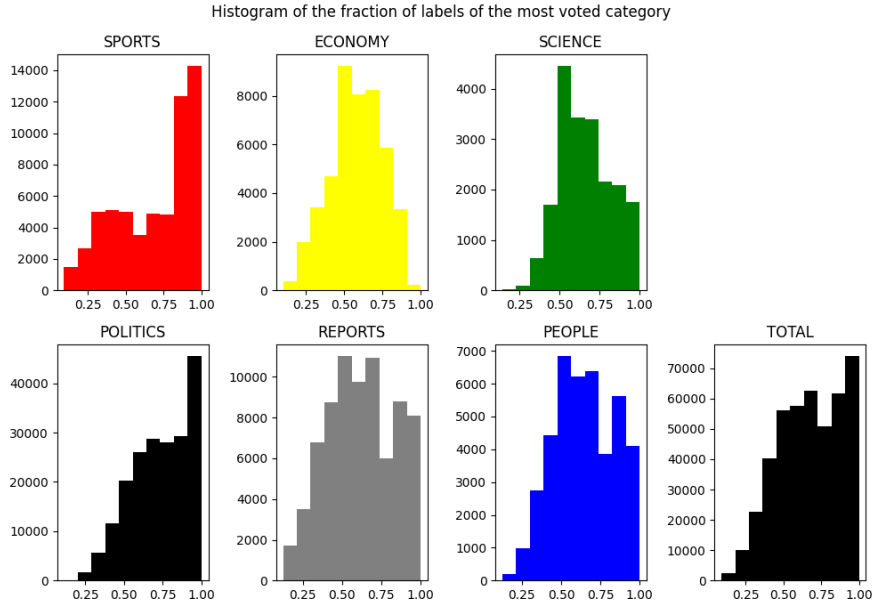


Figure 5.7: Histograms with fractions of the labels of the LFs corresponding to the most voted one (not counting abstentions).

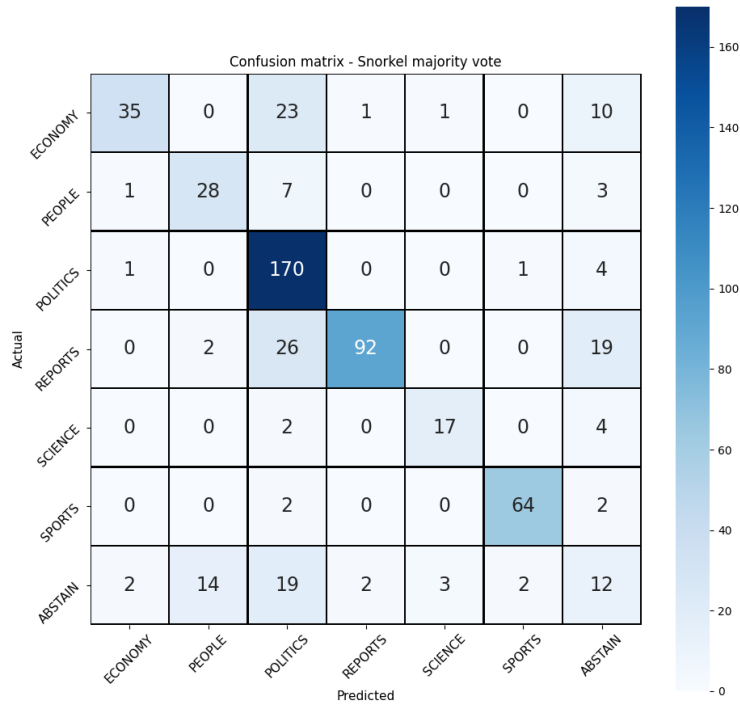


Figure 5.8: Confusion matrix for the Snorkel results with a majority vote strategy

	precision	recall	f1-score
ABSTAIN	0.222	0.222	0.222
ECONOMY	0.897	0.500	0.642
PEOPLE	0.636	0.718	0.675
POLITICS	0.683	0.966	0.800
REPORTS	0.968	0.662	0.786
SCIENCE	0.810	0.739	0.773
SPORTS	0.955	0.941	0.948
accuracy			0.735
macro avg	0.739	0.678	0.692
weighted avg	0.770	0.735	0.730

Table 5.10: Metrics for the Snorkel results with a majority vote strategy

### 5.2.2 Full generative model

As described in section 4.3, implementing the generative model with the results of all the LFs produced for the news a set of probabilities of belonging to each category. The labels to analyze in this section are the ones obtained considering the most probable label for each news item. This strategy only abstains when there is tie, and this usually only happens when all the LFs abstained, which occurs in a very small percentage of the news. The generative model ran for 100 epochs (it means, 100 iterations over the data) and it took about 3 hours and 20 minutes.

The corresponding confusion matrix in figure 5.9 shows a very similar arrangement with the majority vote case. Table 5.11 shows that total accuracy grows 1.4%, taking into account that as there is no abstention all the cases of abstention in the test set are going to be errors, so the maximum accuracy achievable in this case would be 0.905 (assuming 515 correctly classified news out of the 569 that form the test set, and failing the 54 abstentions).

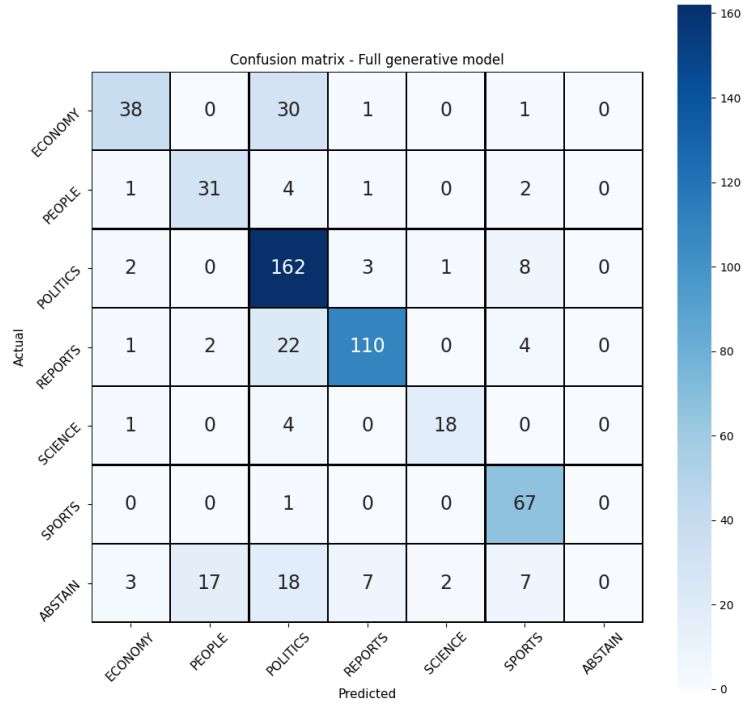


Figure 5.9: Confusion matrix for the Snorkel results with the generative model

	precision	recall	f1-score
ABSTAIN	0.000	0.000	0.000
ECONOMY	0.826	0.543	0.655
PEOPLE	0.620	0.795	0.697
POLITICS	0.672	0.920	0.777
REPORTS	0.902	0.791	0.843
SCIENCE	0.857	0.783	0.818
SPORTS	0.753	0.985	0.854
accuracy			0.749
macro avg	0.661	0.688	0.663
weighted avg	0.697	0.749	0.710

Table 5.11: Metrics for the Snorkel results with the generative model

An interesting analysis before making the final step to obtain the resulting training set at the end of process, is to discover the distribution of probability values of the labels with the maximum probability for each news item introduced in the framework. Figure 5.10 shows this distribution for each category and for the whole set. We can interpret that the news with high probability are the ones where the generative model considers clear the decision. As we see, for most categories this decision is usually very clear, with probability close to 1 in a lot of cases, meanwhile in *economy*, and in a lesser degree also in *science*, this decision is more commonly not so evident.

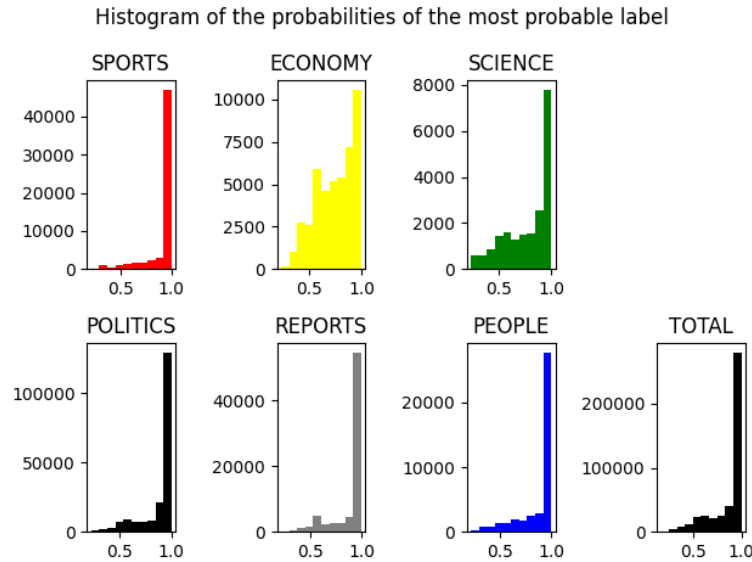


Figure 5.10: Histogram of probabilities of the most probable label after the generative model

It seems reasonable to establish a threshold of probability for considering a news item

into the final train set. This threshold has been selected at 0.8, due to the big amount of news that have a bigger probability than that according to the histograms, so we can achieve a big train set where the news have with high probability the indicated label.

### 5.2.3 Final train set

After the previous filter due to the probability threshold, we obtain a final training set of 331.752 news. As a reminder, the total size of news was of 438.029, so the 75.7% of the initial news will be finally part of the consolidated training corpus. Figure 5.11 shows the distribution and number of news in final train set. Compared with the distribution in the test set (figure 3.7), that we considered to be loyal to the distribution of the dataset, we find a bit of over representation of *politics* category and less *economy* news than expected. This is logical, because the fraction of *economy* labelled news with higher probability than the threshold is smaller than in other categories. One solution could be using different thresholds for each category, with the risk of introducing more wrong or unclear labels to the final dataset.

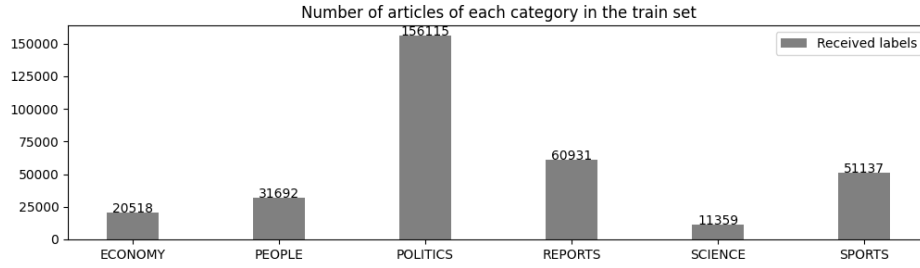


Figure 5.11: Number of news of each category in the final train set

At last, we will explore and evaluate the results of the train set achieved. First of all we have to explain the methodology because for this case it is slightly different. As some of the news in the original set have been filtered, not all the news in the test set are contained in the final set, so those news cannot be evaluated. Therefore, the not selected ones are not taken into account for the evaluation with the test set. The size of the effective test set is reduced from the original size of 569 news to 441 (77%).

In figure 5.12 we can see, first of all, a confusion matrix with similar appearance to the previous ones, again with a clearly highlighted diagonal, where the only not diagonal cells with significant values are the errors with abstention, and the *economy* and *reports* conflicts with *politics*. In metrics table (table 5.13) we can see a total accuracy of 0.82, finding that the filter by probability increased accuracy in 7.5 points, and more than 8.5 in relation to the majority-vote baseline.

For each individual category, we can see that also the f1-score of all them increased,

excepting in the case of the *economy*. We find satisfactory values for most of the metrics, achieving near-to-one values in some cases such as the recall for *sports* and *politics*, or the precision for *reports*. Once again, the worst valued metrics are for *economy*, but we can see almost all the errors are misclassifications with *politics*, so as we have seen, this makes sense with the nature of the dataset and the overlapping over categories. To go a bit deeper analyzing the misclassified news in the test set, table 5.12 shows some examples, showing just their title for brevity, where we can discover that some of the mistakes are understandable because, even if they have considered to be clear in the crowdsourcing process (if not, they would not be part of the test set), they have a clear relation with the label they have received in this process.

There are also 31 out of 441 (7%) news that are labelled in some category and the test set reveals they do not belong to anyone. This is not a big quantity, so we can consider this is not a big problem for us, and even more, even though they not belong to any of the categories, perhaps some of them are closer to the category that have been labelled, so nevertheless they may contribute to the learning process.

Actual	Prediction	Title
ECONOMY	POLITICS	El Consell cree que podrá cerrar 2012 con la deuda a proveedores más baja de su historia
ECONOMY	POLITICS	El Ayuntamiento de Palencia aprueba el Presupuesto de 71,6 millones para 2014, con congelación de tasas
PEOPLE	SPORTS	Rakitic dará el 'sí, quiero' como la infanta Elena
PEOPLE	SPORTS	Bertín Osborne, sobre Cristiano Ronaldo: "No creo que tenga para dos horas de conversación"
REPORTS	POLITICS	El rapero Pablo Hasél es condenado a dos años de cárcel por enaltecimiento del terrorismo
PEOPLE	POLITICS	Gómez: "El mundo del arte se pone de luto al perder a uno de los máximos referentes del flamenco" <sup>2</sup>
SCIENCE	ECONOMY	Ángel Camacho calcula la huella de carbono de aceitunas y mermeladas para ahorrar energía

Table 5.12: Examples of misclassified news in the final train set

As a sum up of this section, we can high mark some points:

- The data programming process achieved a big-size train set that has a satisfactory accuracy.
- Most of the errors are cases of overlapping over categories, or related to the abstention 'category', so this would introduce less confusion in a learning process than evident misclassifications between other categories.

---

<sup>1</sup>Mentions Tomás Gomez, general secretary of PSOE political party in Madrid during 2007-2015.



## 5.2. Evaluation of aggregation of label functions

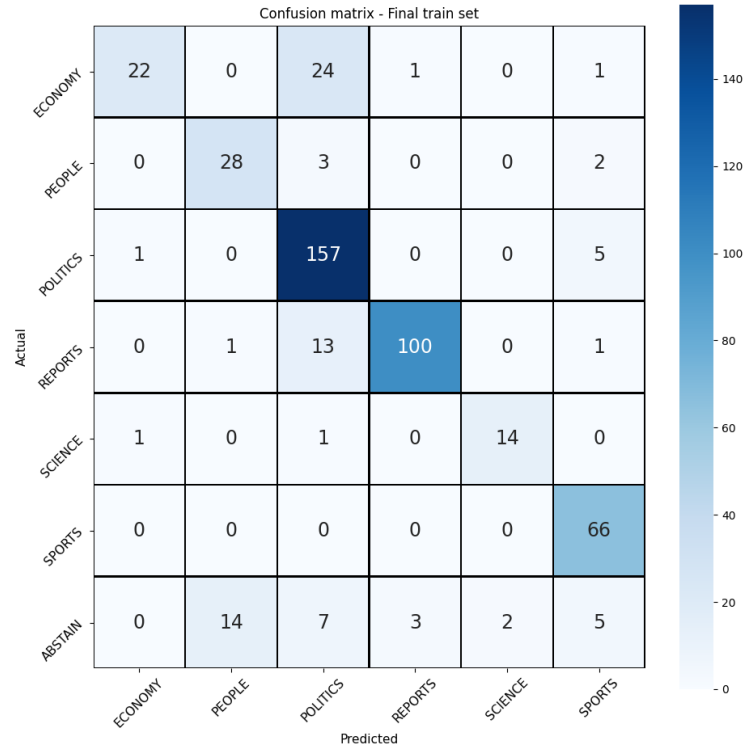


Figure 5.12: Confusion matrix for the final train set results

	precision	recall	f1-score
ABSTAIN	0.000	0.000	0.000
ECONOMY	0.917	0.458	0.611
PEOPLE	0.651	0.848	0.737
POLITICS	0.766	0.963	0.853
REPORTS	0.962	0.870	0.913
SCIENCE	0.875	0.875	0.875
SPORTS	0.825	1.000	0.904
accuracy			0.820
macro avg	0.714	0.716	0.699
weighted avg	0.783	0.820	0.787

Table 5.13: Metrics for the final train set results

### 5.3 Ablation study

This section will describe, explore and analyze the results of the executions of the Snorkel framework with some variations, such as removing some of the LFs, and compare with the results of the full execution. To not fill with figures this section, the confusion matrix and tables of metrics will be located in [Annex I - Results of ablation study](#), and this section will just describe and compare the significant differences found. In general, the results showed will be just the ones of the generative model, to compare with the results of the full generative model in figure 5.9 and table 5.11. Just in punctual cases any other analysis will be made to complete the comparison.

#### 5.3.1 Generative model excluding BERT LF

First, we executed the Snorkel generative model excluding the two BERT based LFs, as explained before, due to the huge amount of time it takes its execution: The difference reduction of time when not using it can vary from about 90 hours to just about 30.

We can see in table 7.1 that, compared to the results of the generative model with all the LFs, total accuracy decreased 2.2 points (0.726 vs 0.749) and the f1-score of the individual categories decreased slightly excepting for *people* and *sports*, where it increased few points. Also it is available the correspondent confusion matrix in figure 7.1.

We can conclude that these LFs provided some useful information, as proved by the reduction in the metrics, but removing them is not determinant, so in cases where the time or the computational resources could be a problem you can get rid of them.

#### 5.3.2 Generative model excluding entities LF

Figure 5.2 showed how the Entities\_LF misclassified a large number of news of all other categories into *politics*. So analyzing now the results of the new execution in figure 7.2 and table 7.2, the first logical consequence of removing this LF is an increase in the precision of the *politics* category, that can be seen in the confusion matrix as smaller values in the column of *politics* predictions, what leads to also a 4 points increase in f1-score. Also in other categories we can see a small increase in f1-score. In overall, the total accuracy is about 1.5 points higher in this case (0.766 vs 0.749), so we can say that the Entities\_LF introduced more noise than useful information for this problem.

Also remember that this LF totally failed with *economy* category, not producing any label of this type. So it is interesting to see how the histogram of the probability of the most probable category (figure 7.3) varies from the original case (figure 5.10), giving this time for *economy* a more similar representation to other categories. Notice that the scale of the plot for *economy* is also different, having this time much more news with high

probability of being of the *economy* category. This agrees with the increase in recall for *economy* we saw in the metrics in the same figure. These improvements probably would solve the under-representation of the *economy* category in the final train set that we identified in the figure 5.11.

### 5.3.3 Generative model excluding monetarian LF

The heuristic LF based on monetarian keywords had really bad metrics for *economy* (Precision: 0.378, Recall: 0.243, F1: 0.296). Removing this LF, as we can see in the metrics in figure 7.4 and table 7.3, does not make a big difference, in fact, *economy* f1-score is reduced (0.637 vs 0.655) and overall accuracy remains almost the same (0.745 vs 0.749). It makes sense that this removal does not affect very much, due to the small coverage this LF had, labelling just the 11.6% of the whole dataset.

## 5.4 Recapitulation of results

Finally, we just show a plot in figure 5.13 comparing the different values of total accuracy of different implementations mentioned during this document, and also the accuracies of the LFs. Note that for LFs that only predict a subset of labels total accuracy metric makes no sense, so they are excluded from this comparison.

We can see there that by aggregating labels with either the majority vote model or the generative model (blue) we achieve a considerable improvement compared to the average accuracy of the LFs (green). Similar results are obtained with the generative models used in the ablation study (also blue), and finally there is the final train set (red), where we can see that the filters of low confidence news improved even more the accuracy.

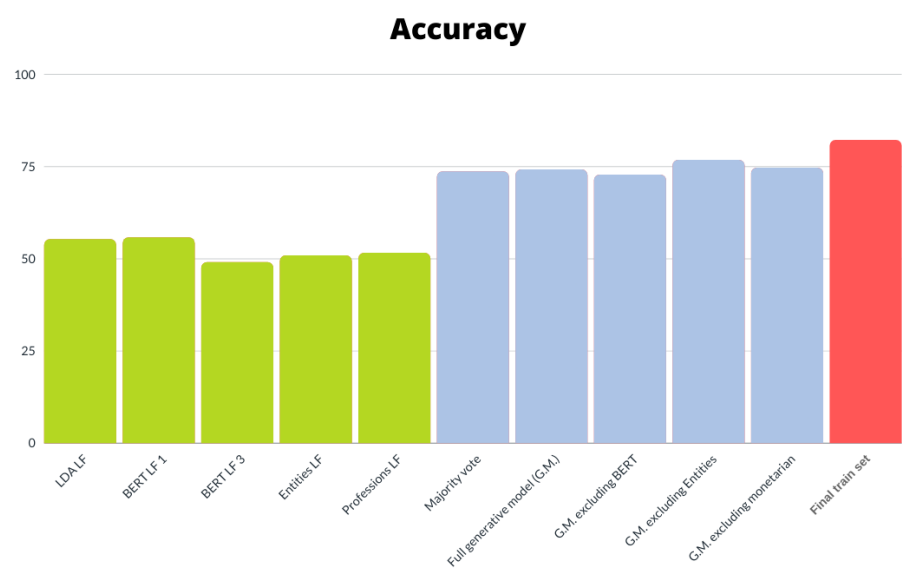


Figure 5.13: Comparative of accuracy results of different implementations.

## 6 Conclusions and future work

### 6.1 Conclusions

In this master thesis we have faced the problem of labelling a large dataset of journalistic news into some determined categories using data programming mechanisms. We have first researched the state of the art of data programming solutions and frameworks, selecting Snorkel as the most suitable tool for our purpose. We have developed a solution to the problem by designing an heterogeneous set of LFs of very different nature, and the combination of their results using Snorkel produced a large labelled dataset of more than 330K texts with much less human effort and time than it would cost obtain it by handlabelling. It is noisy, as expected in datasets labelled with this kind of strategies. The validation results showed an accuracy of 82%, which is a satisfactory value. It has also been discussed how using or not certain LFs that can be conflictive in different ways affects to the results, giving us the knowledge to make decisions in similar situations in the future. With the experience and knowledge obtained we are now qualified to answer the questions proposed in the objectives section:

- *How difficult is to apply data programming to obtain a labelled corpus in real problems?*

Based on the experiment completed, the main difficulties for applying data programming can appear in the definition of a large enough and heterogeneous set of LFs. Some frameworks provide methods for automatically obtaining them. In the case of Snorkel, the LFs are designed by the users, giving them freedom and the possibility of creating LFs of very different complexities. Some of the ones we developed are very simple and intuitive, such as the ones based on keywords or the one consisting of checking names in a list. Others may require expending more time for evaluating their viability and setting parameters (e.g. clustering based LFs such as the LDA based one), or may rely on having at least a small set of labelled points (like the logistic regression classifiers based on TF-IDF representations). The good point is that the LFs explored and designed here can be useful in other text classification

problems with multiple labels, just by making small changes: adapting keywords, obtaining a list of names of representative concepts for the categories (like the case of *sports* for this problem), or readjusting some parameters or thresholds in the executions.

Note that journalistic categories classification is a problem that do not require very specific knowledge of the subject, as the categories are easily recognizable for everybody, but other classification might require more specific knowledge (e.g. medical or other technical issues). This may require the participation of SMEs for designing LFs.

- *How can these mechanisms be used to improve the existing models? For example, updating the training of a model when the context has changed from the original labels.*

It is feasible to use data programming and Snorkel to adapt faster to changes in the context (e.g. a change in some of the original categories of the problem) due to the simplicity of some LFs, and specifically the ones designed for this problem. They can be adapted to new categories just by discovering keywords in a category, or common patterns and regex expressions in them. For some of them, like the BERT-masking based LF or the LDA based, a small exploration of the context and the results of the algorithms (predictions in BERT and common words of LDA topics) must be done for discovering the useful keywords for that LF. So we just have to make those adaptations to the new categories and run again the framework to relabel the data and obtain a new training set to re-train the model. The same process can be done in order to increase the training size when more unlabelled data are available.

- *What problems present these kind of approaches?*

One of the main challenges on these approaches can come from the design of LFs. The implementation of simple LFs sometimes can have limited coverage and more complex ones are needed. One thing to point out is that in the addressed problem we had texts of a considerable length, but for other problems with shorter texts (typical examples can be microblogging texts or product reviews) the coverage of some of the LFs used here should probably be lower. Therefore different strategies for LFs not considered for the solution can have more impact with this type of texts (informal and short). Other problems that we have experienced are the computational complexity that some LFs needed (e.g. those employing deep learning), requiring sometimes large computational resources or time. The difficulty to label some texts correctly due to the overlap in some categories (e.g. with *economy* and *politics* in our problem) or the influence of biases and subjectivity at the time to label were also present in our experiment. However, all of these problems are not specific for these strategies as they are common problems for text classification tasks.

## 6.2 Limitations

In this section we explain some of the limitations faced during the course of the development of this master thesis.

- One of the main limitations is the size of the test set for evaluating the results. 569 news can be considered not a very big size, but an acceptable size. The problem comes with the distribution of labels in the dataset. We saw that some categories such as *science* had less representation in the dataset than average and consequently it only counted with 23 appearances in the test set, which is a small quantity that could not represent well the diversity of texts that can belong to the *science and technology* category. Also it makes that the change of classifying correctly or wrongly one news item will have a significant impact in the metrics.

Related to the incapability for handlabelling more data, we also have an absence of a big handlabelled dataset of the same problem, which prevents us for training a model directly with the handlabelled dataset, for comparing the results with the homologous trained with the data programming generated data. One of the most interesting results will be to get the number of noisy data obtained by data programming needed to achieve the same results with a fully supervised model using the hand labelled data. Studies such as the Snorkel DryBell experiment [10] states that this equivalence can be in a wide range depending on the problem to solve. For example, for a topic classification task, with 80K handlabelled examples they obtained the same accuracy as with 684K examples labelled by Snorkel, meanwhile for a product classification task they needed 12K handlabelled examples to achieve the accuracy of the 6.5 millions of labels generated by Snorkel.

Regardless, for this problem we have seen the different effort needed for handlabelling and for labelling with data programming and we can assure that it will be much more feasible to obtain data with the data programming. In fact, the absence of labelled datasets is very common in practice and here is where data programming techniques are more interesting.

- Other limitation comes from the fact of just make one case of study for discovering the potential and utility of data programming and get conclusions from it. For generalize these conclusions, further experiments must be done, for example using another taxonomy of labels for the same data, using different LFs or using texts of a different nature instead of news.

## 6.3 Ethics

Many classification problems involve ethics questions and biases produced by the training data that may be taken into account during their evaluation. Specially those that use

data involving personal or delicate data such as race, gender, religion, health data, etc., that can lead to models that discriminate by any of these reasons.

The problem addressed in this thesis has little susceptibility to be affected by this due to the, a priori, low impact that the categorization may have, but future uses of the generated dataset may have more delicate applications so it must be taken into account that there are news about all kinds of topics and some of them (violence against women, racial conflict, political opinions, etc.) may, eventually, lead to biases, so a good practice could be study the possible implications of the potential biases and how to mitigate them before using this data for any application.

Other aspect to put attention on is the environmental impact and carbon footprint some deep learning systems may have. Some of the state of the art models for natural language processing such as BERT, GPT-2 [29] or GPT-3 [30] are costly to train both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint produced by the massive use of modern tensor processing hardware (GPUs and TPUs). Influent factors here are the use of renewable energies or the efficiency of the employed hardware. There are studies like [31] and [32] evaluating the cost of some of the previous models and factors.

### 6.4 Future work

There are different aspects that can extend and improve the work done in this master thesis:

- Works related with the improvement of the current solution. One of the problems in the developed LFs is the running time of some LFs (such as those based on computational intensive deep learning models: e.g. BERT). One way to reduce this time is shortening the texts that enter the model, although this implies in discarding useful information that can alter the performance.

We used the masking capabilities of BERT to generate a labelling function. Another approach to a similar LF using BERT is described in [33], where it is used a small piece of text in the input accompanied with a masked prompt to make the model predict a word for that masked space that can be useful for the classification task. A big difference is that they made a previous fine-tuning step with labelled texts and transformed the sentences using the same prompts that can later make the predictions.

Also it can be interesting to experiment with other models, specially with BERT evolutions that are specific for the Spanish language such as BETO [34] or RigoBERTa [35] (this last one is not even available yet at the moment of this writing<sup>1</sup>).

---

<sup>1</sup>May 2021



Other task for improving performance for this problem can be the development of other kind of LFs. Here there are some other ideas that can be extended for this purpose:

- Using a source of information such as Wikipedia to check information about the contents of the texts. For example, search for entities such as names, organizations or institutions and focus for example in the main information in their correspondent Wikipedia page, or in the first paragraph which usually contains the most relevant information. For example, searching for the name of a person could tell you about if they are a politician, a footballer, a scientist, etc.

There are versions of the Wikipedia in different formats that can be downloaded and allow to make quick offline searches of their content, so probably a LF using this would not be very time expensive.

- Similarly to the previous case, semantic technologies and Open Linked Data sources of information (such as DBPedia<sup>2</sup>) can be used to check information about words or entities and automatically discern if its related to any of the defined categories.

- Works related with the automation of the implemented workflow. An important and useful advance would be creating a pipeline for automatizing all the process here developed, where the users just need to provide the data and set up some configuration files indicating the desired categories, keywords for each of them and other resources, parameters or characteristics, and run the corresponding scripts to label their datasets without need of changes in the code.
- Works oriented to simplify the generalization to other problems. Related with the need of generalization exposed in the limitations section, more similar experiments can be done exploring solutions and performance in other problems. First of all, the taxonomy for the solved problem can be changed, introducing new categories (e.g. justice, culture, international issues, etc.) to see if the performance remains similar. Also it can be checked the influence of replacing all the LFs (or a majority of them) by different ones with similar individual performances to assure the correct functioning of the label aggregators. Finally, the experiment must be repeated with different types of texts rather than news, to assure this mechanisms perform well with texts of different natures. For example, as commented before it would be reasonable that the LFs would have less coverage in shorter texts (e.g. product reviews or tweets). So hypothetically, problems with those kind of texts would need more LFs to achieve similar performances.

---

<sup>2</sup><https://es.dbpedia.org/>



## 7 Annex I - Results of ablation study

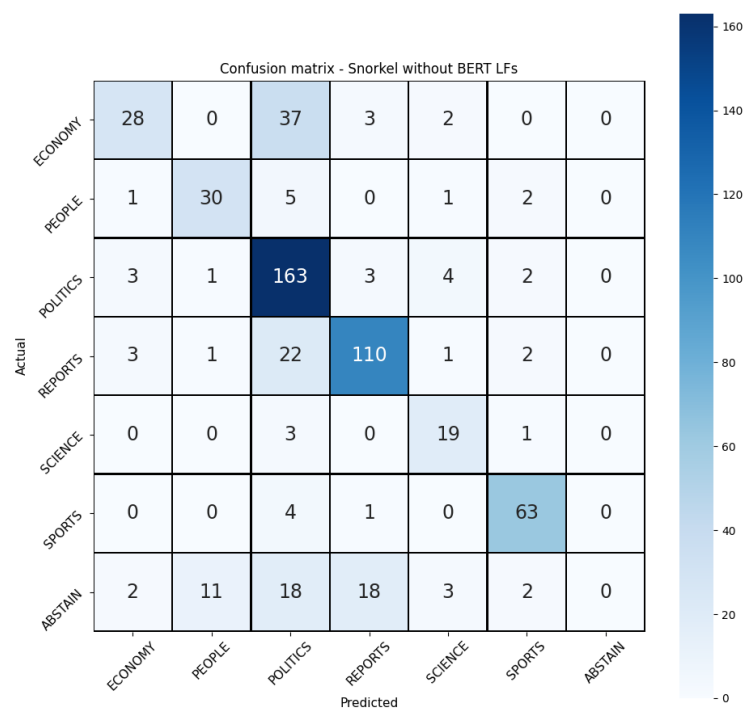


Figure 7.1: Confusion matrix for the Snorkel results with a generative model excluding the BERT based LFs

	precision	recall	f1-score
ABSTAIN	0.000	0.000	0.000
ECONOMY	0.757	0.400	0.523
PEOPLE	0.698	0.769	0.732
POLITICS	0.647	0.926	0.762
REPORTS	0.815	0.791	0.803
SCIENCE	0.633	0.826	0.717
SPORTS	0.875	0.926	0.900
accuracy			0.726
macro avg	0.632	0.663	0.634
weighted avg	0.670	0.726	0.683

Table 7.1: Metrics for the Snorkel results with a generative model excluding the BERT based LFs

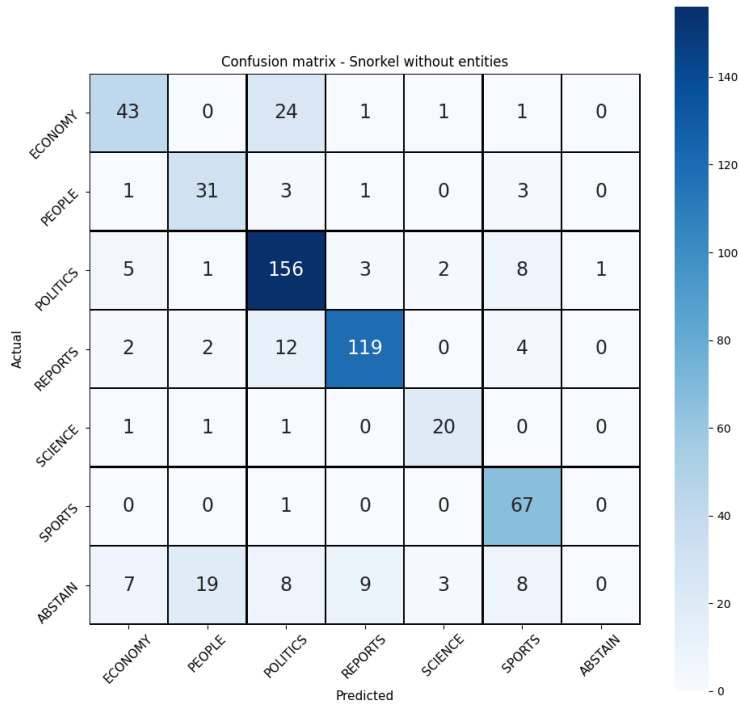


Figure 7.2: Confusion matrix for the Snorkel results with a generative model excluding the entities LF

---

	precision	recall	f1-score
ABSTAIN	0.000	0.000	0.000
ECONOMY	0.729	0.614	0.667
PEOPLE	0.574	0.795	0.667
POLITICS	0.761	0.886	0.819
REPORTS	0.895	0.856	0.875
SCIENCE	0.769	0.870	0.816
SPORTS	0.736	0.985	0.843
accuracy			0.766
macro avg	0.638	0.715	0.669
weighted avg	0.702	0.766	0.728

---

Table 7.2: Metrics for the Snorkel results with a generative model excluding the entities LF

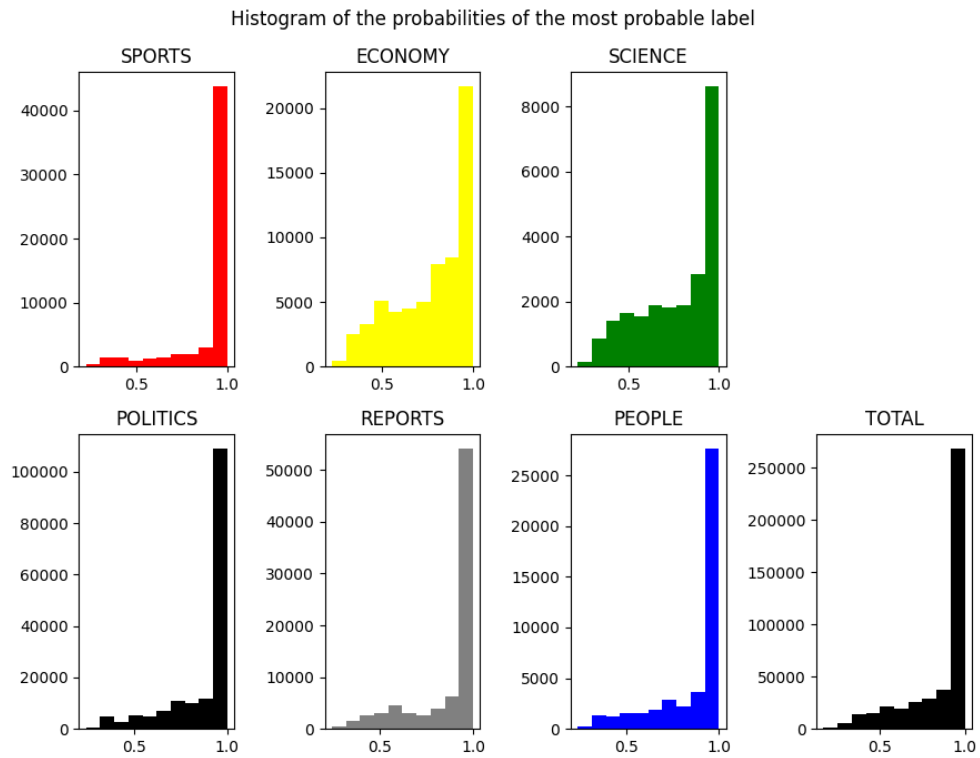


Figure 7.3: Value of probability of the most probable label for each record after excluding the Entities\_LF

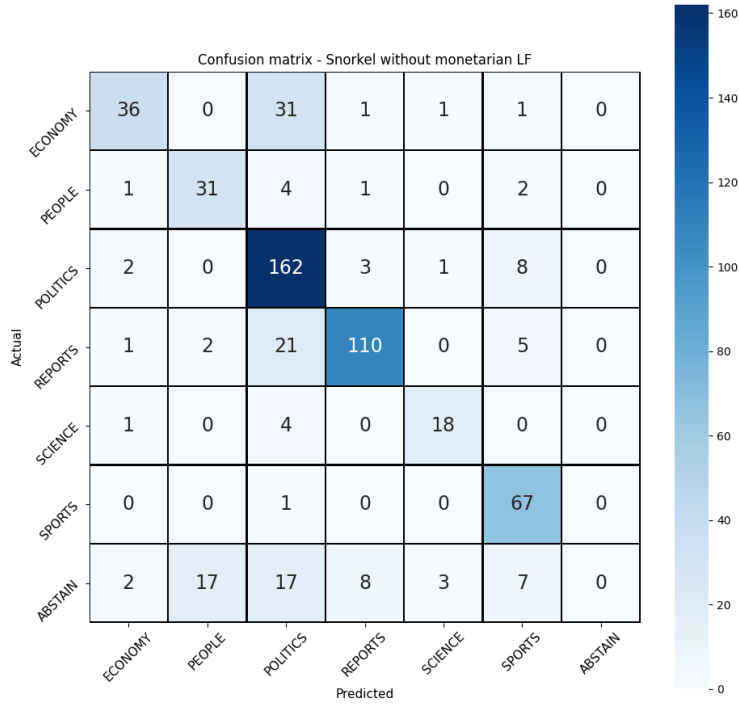


Figure 7.4: Confusion matrix for the Snorkel results with a generative model excluding the monetarian LF

	precision	recall	f1-score
ABSTAIN	0.000	0.000	0.000
ECONOMY	0.837	0.514	0.637
PEOPLE	0.620	0.795	0.697
POLITICS	0.675	0.920	0.779
REPORTS	0.894	0.791	0.840
SCIENCE	0.783	0.783	0.783
SPORTS	0.744	0.985	0.848
accuracy			0.745
macro avg	0.651	0.684	0.655
weighted avg	0.693	0.745	0.705

Table 7.3: Metrics for the Snorkel results with a generative model excluding the monetarian LF

# Bibliography

- [1] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [2] Vikramkumar, B. Vijaykumar, and Trilochan, “Bayes and naive bayes classifier,” *ArXiv*, vol. abs/1404.0933, 2014.
- [3] D. R. Cox, “The regression analysis of binary sequences,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.
- [4] “Neural networks,” <https://www.ibm.com/cloud/learn/neural-networks>, last accessed: 29-05-2021.
- [5] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Essen, A. Awwal, and V. Asari, “A state-of-the-art survey on deep learning theory and architectures,” *Electronics*, vol. 8, p. 292, 03 2019.
- [6] A. Ratner, P. Varma, C. Ré, and B. Hancock, “Weak supervision: A new programming paradigm for machine learning,” <https://ai.stanford.edu/blog/weak-supervision/>, last accessed: 21-05-2021.
- [7] A. Ratner, C. De Sa, S. Wu, D. Selsam, and C. Ré, “Data programming: Creating large training sets, quickly,” *Advances in neural information processing systems*, vol. 29, p. 3567, 2016.
- [8] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: rapid training data creation with weak supervision,” *Proceedings of the VLDB Endowment*, vol. 11, no. 3, p. 269–282, Nov 2017. [Online]. Available: <http://dx.doi.org/10.14778/3157794.3157797>
- [9] A. Amidi and S. Amidi, “Supervised learning cheatsheet,” <https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning>, last accessed: 14-06-2021.
- [10] S. H. Bach, D. Rodriguez, Y. Liu, C. Luo, H. Shao, C. Xia, S. Sen, A. Ratner, B. Hancock, H. Alborzi *et al.*, “Snorkel drybell: A case study in deploying weak supervision at industrial scale,” in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 362–375.

## Bibliography

---

- [11] P. Varma and C. Ré, “Snuba: Automating weak supervision to label training data,” *Proceedings of the VLDB Endowment*, vol. 12, pp. 223–236, 11 2018.
- [12] “Automating weak supervision,” <https://blog.fastforwardlabs.com/2019/09/27/automating-weak-supervision.html>, last accessed: 26-05-2021.
- [13] B. Hancock, M. Bringmann, P. Varma, P. Liang, S. Wang, and C. Ré, “Training classifiers with natural language explanations,” in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2018. NIH Public Access, 2018, p. 1884.
- [14] D. Fu, M. Chen, F. Sala, S. Hooper, K. Fatahalian, and C. Ré, “Fast and three-rious: Speeding up weak supervision with triplet methods,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 3280–3291.
- [15] P. Tamagnini and A. Nembach, “Weak supervision deployed via guided analytics,” <https://www.knime.com/blog/weak-supervision-guided-analytics>, last accessed: 26-05-2021.
- [16] P. Varma, B. He, D. Iter, P. Xu, R. Yu, C. De Sa, and C. Ré, “Socratic learning: Augmenting generative models to incorporate latent subsets in training data,” *arXiv preprint arXiv:1610.08123*, 2016.
- [17] F. Duplessis, S. Chow, and S. Prince, “Generating labels for model training using weak supervision,” <https://www.borealisai.com/en/blog/generating-labels-model-training-using-weak-supervision/>, last accessed: 25-05-2021.
- [18] V. Yadav and S. Bethard, “A survey on recent advances in named entity recognition from deep learning models,” in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2145–2158.
- [19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [21] W. Scott, “Tf-idf from scratch in python on real world dataset,” <https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089>, last accessed: 26-05-2021.
- [22] D. Blei, A. Ng, and M. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 05 2003.



- 
- [23] A. Ram, J. Sunita, A. Jalal, and K. Manoj, “A density based algorithm for discovering density varied clusters in large spatial databases,” *International Journal of Computer Applications*, vol. 3, 06 2010.
- [24] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “Optics: Ordering points to identify the clustering structure,” *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.
- [25] P. Gamallo, M. Garcia, C. Pineiro, R. Martinez-Castano, and J. C. Pichel, “Linguakit: a big data-based multilingual tool for linguistic analysis and information extraction,” in *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2018, pp. 239–244.
- [26] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, “Stanza: A python natural language processing toolkit for many human languages,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Jul. 2020, pp. 101–108. [Online]. Available: <https://www.aclweb.org/anthology/2020.acl-demos.14>
- [27] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” *arXiv preprint arXiv:1910.01108*, 2019.
- [28] “Taking snorkel for a spin,” <https://blog.fastforwardlabs.com/2019/07/08/taking-snorkel-for-a-spin.html>, last accessed: 24-05-2021.
- [29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [30] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [31] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for deep learning in NLP,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3645–3650. [Online]. Available: <https://www.aclweb.org/anthology/P19-1355>
- [32] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, “Quantifying the carbon emissions of machine learning,” *arXiv preprint arXiv:1910.09700*, 2019.
- [33] T. Schick and H. Schütze, “Exploiting cloze questions for few-shot text classification and natural language inference,” *arXiv preprint arXiv:2001.07676*, 2020.
- [34] J. Canete, G. Chaperon, R. Fuentes, and J. Pérez, “Spanish pre-trained bert model and evaluation data,” *PML4DC at ICLR*, vol. 2020, 2020.

## Bibliography

---

- [35] “Nuevo modelo de lenguaje para pln en big things 2020,” <https://www.iic.uam.es/noticias/nuevo-modelo-lenguaje-pln-en-big-things-2020/>, last accessed: 25-05-2021.