FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Suggesting Human Resources for Project Tasks

Ana Sá e Sousa Carneiro da Silva

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Pedro Mendes Moreira

Second Supervisor: Filipe Figueiredo Correia

October 31, 2021

Suggesting Human Resources for Project Tasks

Ana Sá e Sousa Carneiro da Silva

Mestrado Integrado em Engenharia Informática e Computação

October 31, 2021

Abstract

Human resource allocation can determine the success or failure of an organization due to the impact that it can have on its performance, costs, and team motivation. Allocating the right person for a task dictates how it is made, its quality, and the time spent accomplishing it. Therefore, the person in charge of designating which employees of an organization will be responsible for which task has a great responsibility and demanding assignment. Furthermore, it is also a very complex job due to the restrictions involved in matching a person and an assignment. One must consider which knowledge is required to carry it out, which person has the most appropriate experience, how many people are required, how much time it will take, and if those people are available, among others. An automatic triage can be helpful in this decision-making process.

The present dissertation studies different Machine Learning approaches to recommend people for a given task. The goal is to match the data that describes a person's skills with the one that characterizes the task. A key challenge of using machine learning in these problems is how to choose which data to use. In fact, it is not trivial to decide whether or not the data regarding each person or each task should be included and how it should be linked. Two data construction methods are compared in order to find the best approach for this issue. One relies on each person's task history. Hence, the classifier's job is to understand which type of tasks the person has performed before and whether or not the current task is similar to those. The other is essentially the Cartesian Product between all the tasks and people's information. In this case, the model must try to extract some correlation between the two entities' data.

SVM and Naive Bayes classifiers were used in the experiments. Surprisingly, the latter achieved the best results due to the inherent imbalance nature of the problem and the SVM's incapacity to deal with it properly. Furthermore, given the problem's similarities to Entity Matching, some of its techniques were adapted. Namely, a blocking method was used as well as a predictor that correlates both entities.

In every experiment, the classifier computed each person's probability to do a task. The people with the highest values were recommended. The classifier's performance was evaluated using metrics specific to an imbalanced data set. Moreover, the recommendations provided by the approach were judged based on metrics of multi-label problems. The best approaches were able to reduce the possible number of people by 50%, achieving an accuracy of 0.2 and 0.1 for the classifier's performance and the experiment as a whole, respectively.

The present dissertation tackles the people allocation problem whose optimization is important to any company and has not been studied for the industry domain. The problem's definition, the experiments implemented, and the proposed ideas can serve as a foundation for future solutions.

Keywords: Machine Learning, Task Allocation, Imbalanced Classification, Predictive Modeling, Multi-Label Classification

Resumo

O modo como os recursos humanos são distribuídos pode ditar o sucesso ou falhanço de uma organização devido ao impacto que pode exercer na qualidade de execução, nos custos e na motivação da equipa. Alocar a pessoa correta a uma tarefa determina a sua realização, influenciando aspetos como a qualidade, o tempo gasto, entre outros. Assim, a pessoa encarregue por decidir quem ficará responsável por que tarefa tem uma grande responsabilidade e um cargo exigente. É um trabalho muito complexo devido às inúmeras restrições que se devem considerar quando se faz corresponder uma tarefa a uma pessoa. É fundamental saber qual o conhecimento necessário para completar a tarefa, que pessoas têm o perfil mais adequado, quanto tempo tomará dos funcionários e quem estará disponível na alturar em questão. Uma triagem automática é extremamente útil neste processo de decisão de forma a diminuir o número de possíveis erros.

Esta dissertação estuda diferentes métodos de Machine Learning para recomendar pessoas para uma dada tarefa. O objetivo é corresponder informação representativa do perfil e competências de uma pessoa a dados que caracterizam uma tarefa. Um desafio a ter em conta no uso de Machine Learning nestes problemas é a escolha dos atributos que devem ser usados. Não é um problema trivial decidir se os dados da pessoa, os da tarefa ou ambos devem ser tidos em conta e como os relacionar. Dois métodos de data construction foram comparados de forma a identificar o mais apropriado. Um deles depende do histório de tarefas a que cada pessoa já esteve alocada. Deste modo, o classificador tem de identificar o tipo de encargos já realizados por cada um e se a tarefa atual se enquadra ou não nesse grupo. O segundo método corresponde ao produto cartesiano entre todas as tarefas e todas as pessoas. Neste, o modelo tem de tentar extrair alguma correlação entre as duas entidades em questão.

Os classificadores SVM e Naive Bayes foram os usados nas experiências. Surpreendentement, este último foi o que obteve melhores resultados divido à incapacidade do SVM lidar com dados desbalanceados, que caracterizam problemas como este. Para além disso, divido à semelhança da questão em estudo com Entity Matching, algumas técnicas foram adaptadas. Nomeadamente, métodos de blocking foram usados tal como atributos que relacionam as duas entidades.

Em cada experiência corrida, o classificador indicava a probabilidade de uma pessoa realizar uma tarefa e as pessoas com maior probabilidade eram recomendadas. Os resultados obtidos foram avaliados usando métricas específicas para dados desbalanceados. Além disso, as recomendações obtidas foram analisadas com métricas de problemas de multi-label. A melhor abordagem conseguiram reduzir o número de pessoas para 50%, obtendo uma accuracy de 0.2 nos resultados do classificador e 0.1 nos resultados da experiência como um todo.

Este trabalho debruça-se sobre o problema de alocação de tarefas cuja otimização é importante para qualquer organização e não foi estudado para o domínio industrial. A definição do problema, as experiências implementadas e as ideias propostas servem de base para futuras soluções.

Keywords: Machine Learning, Dados desbalanceados, Classificação Multi-Label

Acknowledgements

Throughout the writing of this dissertation, I have been lucky to receive a great deal of support and assistance. Many people contributed to this project in a direct or indirect manner. Without them, writing this thesis would have been far more challenging.

Firstly, I would like to thank my supervisors, Professor João Pedro Mendes Moreira and Professor Filipe Correia, for all the patience, constructive guidance, and availability throughout the planning and development of this research work. I would also like to thank Ricardo Graça for providing me all the needed data as promptly as possible and being extremely available to answer any doubts.

I am also extremely grateful to family and friends for always cheering me up and making me believe I could see this through. Thank you for calming me down in more stressed times, for putting up with my crazy talks, and for helping me become the person I am today.

Thank you,

Ana Silva

"Of all things the measure is man"

Protagoras of Abdera

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
AUC	Area Under the ROC curve
BoW	Bag of Words
DL	Deep Learning
DT	Decision Tree
EM	Entity Matching
FN	False Negatives
FP	False Positives
HRA	Human Resource Allocation
IDF	Inverse Document Frequency
IR	Information Retrieval
KNN	K-Nearest Neighbours
LC	Label Cardinality
LD	Label Density
ML	Machine Learning
MNB	Multinomial Naive Bayes
NB	Naive Bayes
NLP	Natural Language Processing
PM	Project Manager
PROMESSA	PROject Management Intelligent aSSistAnt
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TF	Term Frequency
TN	True Negatives
TP	True Positives

Chapter 1

Introduction

Human resource allocation (HRA) can determine the success or failure of an organization. HRA can impact a company's performance, costs, and team motivation. Allocating the right person for a task dictates how its quality, the time spent accomplishing it, and the enthusiasm put into finishing it [3]. A mistake in assigning a person for a job can lead to delays or even errors which can mean financial losses. The allocation is a crucial job, and it is usually the responsibility of the Project Managers (PMs), who often do not have all the necessary information to do it properly. There are many restrictions involved: one must think about which knowledge is required by the task, which person has the most appropriate experience, how much time it will take, how many people are needed, and if they have enough schedule. Even if all of that data is stored and available, it is too much information to be aware of and make the best decision. Moreover, the bigger an organization is, the more people there are, the more data to be aware which makes it almost impossible. Another aspect to take into account is the subjectivity that plays into all human decisions [32]. For example, if PMs are accustomed to working with someone, they will probably choose them even if they are not the best match for the job in question. Given the importance of the decision, it should be done as impartial as possible.

1.1 Motivation

In their study, Barcus et al. [4] concluded that unsupported decision making could lead to poor quality decisions, struggles in the planning of recurrent decisions, shortage of information sharing, and unpredictable performances. They also deduced that it is impossible to centralize all the needed data in a person or even a group. Therefore, it is necessary to triage the people for a given task and make the decision process faster and more accurate.

Machine learning techniques proved to be a great solution for automatizing processes in several domains such as software for the automotive industry, image recognition, software cost estimation, and industrial production assembly lines. In the software engineering sphere, ML has

been used to assign bugs and pull requests reviews automatically.

The present dissertation inserts itself in PROMESSA[1] (PROject ManagEment Intelligent aSSistAnt), a project that aims to develop a complex system that incorporates several project management tools that support project managers in their tasks and decision-making processes. It is supposed to analyze all the information in order to predict risks and consequences, or to forecast the completion of a set of tasks, and make relevant recommendations [29, 26]. The PROMESSA project is to be integrated and validated with three project management solutions: Project Control, SCRAIM[2] and JIRA[3].

The human allocation problem inserted in the industry domain is very different from the ones found in the literature that usually regard open software or crowd sourcing subjects. These tasks diverge from those present in the industry domain because of the social component (reflected on comments on pull requests or bugs), several people solving the same problems, and a lot of unstructured data such as reports, descriptions, and code files. The differences in the available data and its structure significantly impact the issues that arise and hence the most appropriate solution.

1.2 Objectives

The integration of this feature in the PROMESSA project would help PMs decide which of the company's employees are the most appropriate for a task. The triage of people and recommendation of a subset of candidates could improve this process's accuracy and speed, a great addition to the platform. Therefore, this dissertation aims to explore the task assignment issue for the industry domain, namely for the platform in question, stating all the complications of tackling this problem and possible solutions for them. Furthermore, it intends to propose a viable approach that handles these difficulties with different machine learning methods and, given a task, recommends some people to solve it.

1.3 Hypothesis and Research Questions

The present dissertation aims at testing the following hypothesis:

"Machine learning methods can be applied for task assignment and lead to meaningful results that help triage appropriate people for a given task"

Two concepts must be defined in order to better understand this hypothesis - what are meaningful results and appropriate people. The results are considered meaningful if better than the established baseline according to common evaluation metrics in Machine Learning methods based on the Literature Review. A person is said to be appropriate for a task if it was chosen to do it according to the data provided to test the approach.

[1] https://www.aicos.fraunhofer.pt/en/our_work/projects/promessa.html
[2] https://www.scraim.com/
[3] https://www.atlassian.com/software/jira

This hypothesis is decomposed in the following research questions:

- **Research Question 1** - What are the best classifiers for the task assignment problem in the industry domain?
- **Research Question 2** - What are the best methods for data construction when relating two entities?
- **Research Question 3** - What are the main issues of this problem and which methods can be applied to improve it?

The first question is the most important to answer, as it establishes the success of machine learning and NLP methods to recommend people for a given task. The second question regards how the data of the two entities involved (the people and the assignments) should be related in order to have the best results. The last question concerns the recommendation pipeline in general and attempts to systematize what aspects should be considered when approaching these types of problems.

1.4 Structure

The dissertation is organized into seven chapters whose content and purpose are detailed below.

This initial chapter, chapter 1, introduces the project's context and motivation by stating its importance and relevance nowadays. It also presents this dissertation's hypothesis and the research questions the present work tries to answer.

Chapter 2 explains some background concepts regarding Machine Learning, Natural Language Processing, and Entity matching that help better understand the remainders of the document. The classification methods used are briefly explained as well as NLP pre-processing steps (tokenization, stop words removal, lemmatization), different representation for unstructured data (BoW, TF-IDF) and similarity measures (cosine similarity). Moreover, some evaluation metrics are also explained so the results can be correctly interpreted. Given the problem in question, both the most common metrics and the ones specific for imbalanced datasets are presented.

Chapter 3 analysis the state of the art regarding the match between tasks and people. It divided the works into two sections depending on the domain they insert themselves - open-source software and crowdsourcing tasks. The most commonly used algorithms and evaluation metrics are presented, which helps delineate this dissertation approach and baseline.

In chapter 4 the problem is described and defined as a multi-label problem. A detailed study of the data is provided, namely, its characteristics and how they influence the problem in question. The approaches are briefly described as well as the data construction methods.

Chapter 5 details the steps needed for each approach to take place, namely, describing the resources used, the restrictions imposed, and the decisions that were made.

Chapter 6 presents the results obtained from the different approaches chosen and a discussion comparing each of them and explaining their meaning.

Finally, in chapter 7 the conclusions of this dissertation are drawn, and some future work that improves these works results is suggested.

Chapter 2

Machine Learning approaches and techniques

This chapter briefly presents relevant Machine Learning Techniques that help understand this work's contributions. It introduces machine learning classification, describes the reasoning behind the classifiers (SVM and Naive Bayes) used in this work, and justifies the choice. It also presents the multi-class and multi-label problems and two approaches that are usually used. Afterward, it presents Natural Language Processing key concepts, their importance for this work, the necessary pre-processing steps to handle unstructured data like tokenization, lemmatization, stop word removal, and the ways a text can be represented and used as input for the predictive models, such as BoW and TF-IDF. It also introduces a standard similarity measure - the cosine similarity. Moreover, some relevant Entity Matching concepts and techniques are described. Finally, it details essential evaluation metrics, showing how they are computed and their meaning. There is a focus on metrics specific to imbalanced datasets and the multi-label problem due to the present work's characteristics.

2.1 Machine Learning Classification

Machine Learning is the field of study of computer systems that can automatically improve their performance when processing previous experience [10]. It is a vast subject with numerous applications and methods that have been spreading to more and more domains through the years.

In classification problems, the goal is to process labeled data and from it be able to predict to which class an instance belongs. Several algorithms and methods are commonly used, such as SVMs, KNNs, Naive Bayes (NB), Decision Trees (DT), Artificial Neural Networks (ANNs), among others. The ones chosen for the present work were SVMs and Naive Bayes.

2.1.1 Multi Labels Problems

With the growth and adaptation of classification methods to several domains, it became apparent that not all problems are binary, meaning they cannot be defined with only two classes. Furthermore, even in others, the classes are not exclusive.

There are two different types of classifications depending on the number of labels per instance, *single-label* and *multi-label*. Each instance is only associated with one label in the former, whereas in the latter, an instance can have several labels. An example of a multi-label problem is the assignment of genres to movies. A movie can have several categories simultaneously, like the Lord of the Rings is an action, adventure, and drama movie [7, 35].

Moreover, there are two types of problems that regard the number of existent classes in general, *binary-class* and *multi-class*. In the former, as the name suggests, an instance can only be classified as belonging to two different groups, whereas in the latter, there are three or more classes. The previously example is also a *multi-class* problem because there are more than two genres to assign to a movie [7, 35].

The present section focuses on multi-label problems, given their relevance to this dissertation. The two most common approaches when dealing with these problems are problem transformation methods, and algorithms adaptation methods [36]. The former are algorithm independent and, as the name suggest, they transform the data in order to represent the problem as *single-label multi-class* one or even a *single-label* and *single-class* one. There are several transformations, such as considering only one label and trying to predict whether or not an instance corresponds to a label, creating a unique label for each set of existent multi-labels, among others. The algorithms adaptation methods correspond to changes made in the classifiers so that they support multi-label algorithms. Some studies have developed them successfully by creating AdaBoost.MH and AdaBoost.MR [31] which are extensions of AdaBoost for multi-label data, or ML-KNN [44] based on the KNN classifier, among others.

Another important aspect to correctly characterize the data for *multi-label* problems is the relationship between the instances and the label. The number of labels per entity can impact the performance of the model. Tsoumakas et al. [35] introduces the concepts of label cardinality and label density. Label cardinality (LC) is the average number of labels per instance. Label density (LD) corresponds to the average per instance of the ratio between the number of labels in each example and the total number of labels. Formally, let D be a multi-label data set consisting of $|D|$ instances (x_i, Y_i), $i = 1...|D|$ where x_i are the predictors of the example i and Y are the set of labels associated to it. Equations 2.1 and 2.2 are used to compute the label cardinality and density, respectively. As can be concluded, LC is independent of the total number labels instead of LD, which is not. With the growth and adaptation of classification methods to several domains, it became apparent that not all problems are binary, meaning they cannot be defined with only two classes. Furthermore, even in others, the classes are not exclusive.

There are two different types of classifications depending on the number of labels per instance, *single-label* and *multi-label*. Each instance is only associated with one label in the former, whereas

in the latter, an instance can have several labels. An example of a multi-label problem is the assignment of genres to movies. A movie can have several categories simultaneously, like the Lord of the Rings is an action, adventure, and drama movie [7, 35].

Moreover, there are two types of problems that regard the number of existent classes in general, *binary-class* and *multi-class*. In the former, as the name suggests, an instance can only be classified as belonging to two different groups, whereas in the latter, there are three or more classes. The previously example is also a *multi-class* problem because there are more than two genres to assign to a movie [7, 35].

The present section focuses on multi-label problems, given their relevance to this dissertation. The two most common approaches when dealing with these problems are problem transformation methods, and algorithms adaptation methods [36]. The former are algorithm independent and, as the name suggest, they transform the data in order to represent the problem as *single-label multi-class* one or even a *single-label* and *single-class* one. There are several transformations, such as considering only one label and trying to predict whether or not an instance corresponds to a label, creating a unique label for each set of existent multi-labels, among others. The algorithms adaptation methods correspond to changes made in the classifiers so that they support multi-label algorithms. Some studies have developed them successfully by creating AdaBoost.MH and AdaBoost.MR [31] which are extensions of AdaBoost for multi-label data, or ML-KNN [44] based on the KNN classifier, among others.

Another important aspect to correctly characterize the data for *multi-label* problems is the relationship between the instances and the label. The number of labels per entity can impact the performance of the model. Tsoumakas et al. [35] introduces the concepts of label cardinality and label density. Label cardinality (LC) is the average number of labels per instance. Label density (LD) corresponds to the average per instance of the ratio between the number of labels in each example and the total number of labels. Formally, let D be a multi-label data set consisting of $|D|$ instances (x_i, Y_i), $i = 1...|D|$ where x_i are the predictors of the example i and Y are the set of labels associated to it. And let L be the set of available labels. Equations 2.1 and 2.2 are used to compute the label cardinality and density, respectively. As can be concluded, LC is independent of the total number labels instead of LD, which is not.

$$LC(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} |Y_i| \tag{2.1}$$

$$LD(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i|}{|L|} \tag{2.2}$$

2.1.2 Support Vector Machine

A Support Vector Machine (SVM) is a supervised learning model that represents the instances as points in space so that examples of different classes are divided. The line that separates them is called the hyperplane, and the goal is to find the largest margin that separates them. This margin

is defined by the distance between the instances of both sides that are closest to the border, the so-called support vectors.

Figure 2.1 shows a 2-dimensional representation of instances from both classes (one represented by crosses and the other depicted as circles), their support vectors that are the instances surrounded by a grey square, and the optimal margin and hyperplane that separates them. Other lines that would also split the classes could be drawn, but their distances to all the support vectors would not be as large. Hence they are not optimal.

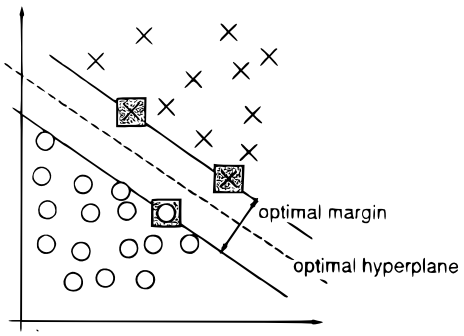

Figure 2.1: Representation of the separation problem [11]

The SVM has been widely used for text classification [18] and it is the one chosen for many studies in the literature review.

2.1.3 Naive Bayes

The Naive Bayes (NB) is also a standard classifier used in text classification. Its name, naive, is due to the simplifying assumptions it makes regarding the independence of attribute values in the test instances. In other words, it assumes that each feature or attribute of the testing data is independent of the others. This is rarely the case, but the classifier still performs well in most problems [1]. It is a good classifier to handle high dimensional data given that it is computationally inexpensive and needs a low amount of memory [18].

The NB bases itself on the Bayes theorem, which describes an event's probability based on prior happenings that influence the event in question. It is represented by the Equation 2.3.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.3}$$

The Naive Bayes version commonly used for text categorization is Multinomial Naive Bayes (MNB) [14].

2.2 Natural Language Processing

Natural Language Processing (NLP) is a field of Artificial Intelligence (AI) that has evolved over the years. Broadly, it can be defined as the analysis and extraction of knowledge by computers from human-readable text. It differs itself from computational linguistics, a very similar field,

due to its focus. While the former aims to design and develop the algorithms responsible for extracting information, the latter's object is to study the language. It only uses computational resources and methods as a way to reach the primary goal. NLP has multiple applications such as Text Classification, Sentiment Analysis, Chatbots, Virtual Assistants, Machine Translation, Text Summarization, among many others.

Natural language is characterized by its lack of organization which makes it hard for machines to process. Therefore, the first step of every approach when dealing with unstructured text is to normalize it and discard irrelevant information. There are several crucial methods that almost all frameworks include, such as tokenization, stop word removal, stemming, and lemmatization.

After filtering out the irrelevant words and standardizing the rest, it is crucial to find an appropriate numeric representation to input the model. There are several possible representations. Ones take into account the times a word, or a letter appears. Others can represent the words' semantic. The most commonly used approaches, such as BoW and TF-IDF are presented below.

Finally, another important aspect of NLP is the measurement of similarity between the representations. Therefore, the cosine similarity computation is also described.

Tokenization

Manning et al [23] defines tokenization as the process of dividing a character sequence into document units. These units can be paragraphs, sentences, letters, or, most commonly, words and will be then called a token. Then, a token is defined as a sequence of characters that are grouped as a useful semantic unit for processing [23].

Tokenization presents some challenges on how to decide words boundaries. Although it is quite an intuitive task for people, computers found some cases a bit ambiguous. Some city names have two words, as *New York*, which could be considered separate tokens or the same one. Some punctuation could be discarded, like commas and full stops, but others, as slashes when separating date numbers, might be important [1].

Furthermore, it is a language-specific problem because words boundaries differ according to the language. For example, words in Chinese are not separated by white spaces or any other characters, unlike English words [12]. Several techniques can be used to split the sequence (splitting using white spaces), and one must be aware of the language's characteristics to choose the best approach.

Figure 2.2 shows an example of tokenization for the sentence *Friends, Romans, Countrymen, lend me your ears;* - all words are separated, and the commas are discarded.

Input: Friends, Romans, Countrymen, lend me your ears;
Output: | Friends | Romans | Countrymen | lend | me | your | ears |

Figure 2.2: Example of tokenization [23]

Stop Word Removal

Not all the words in a text are relevant or convey any specific meaning. However, computers do not have a way to understand which should be given more importance. Additionally, the more words a computer deals with, the less efficient it is. Therefore, it is vital to remove stop words.

Stop words appear many times in a text but are of the least importance. Prepositions and pronouns such as *the*, *and* and *a* are common examples of stop words that don't add any relevant information. Its removal is done by deleting all the tokens that match a word in a stop list (a list where all the stop words are stated) [12, 23]. Stop word removal is also a language-specific problem because frequent words differ between languages.

Sometimes, removing stop words might reduce the information available in a way that harms the system's accuracy. Therefore, some authors prefer to lower the weight given to words that appear many times in the document [1].

Stemming and Lemmatization

Most words can have different forms depending on their grammar type or the sentence's inflection regarding tense or subjects. For example, the noun *weakness* can be an adjective, *weak*, a verb, *weaken*, or an adverb, *weakly* and even as a verb or an adjective can different inflexions, *weakens*, *weakened*, *weakening*, *weaker*, *weakest*. The meaning of two sentences will not be contrasting if one form is chosen over. If every variation is saved as a different token, the computer will handle them as completely different words. That is why stemming and lemmatization are important.

Stemming and lemmatization aim at reducing a word to its basic form by discarding inflections. The former does it by ignoring the affixes while the latter takes into account vocabulary and a morphological analysis [23].

Bag of Words

The Bag of Words model (BoW) is one of the ways to represent a text through a numeric vector. It only takes into account the words used and the number of times they appear, discarding the order they appear on [12]. It starts by computing the vocabulary of the texts, meaning the set of unique words that belong to the texts. Then, each instance will be represented as a vector whose values correspond to one of the vocabulary words and are equal to the number of times that word appears in the instance.

(S1) John likes to watch movies. Mary likes movies too.
(S2) Mary likes to watch games.

For example, considering the sentences above, the vocabulary is the set {a, million, dollars, is, not, cool, do, you, know, what, billion} and each sentence is represented by the vectors shown in Table 2.1.

Sentence	John	likes	to	watch	movies	Mary	too	games
S1	1	2	1	1	2	1	1	0
S2	0	1	1	1	0	1	0	1

Table 2.1: Output of BoW for each sentence

BoW is a language-dependent representation. If two words of different languages have the same meaning, there will be two entrances on both the vocabulary and the vectors that characterize the same word.

Term Frequency - Inverse Document Frequency

Term frequency-inverse document frequency, or TF-IDF for short, is a measure computed for each term or word. It considers the number of times a word or sequence appears in an instance and the number of times that word appears in all the documents. It follows the intuition that the more documents a word is in, the less important it is to describe any of the instances [23]

The TF-IDF measure is a composition of two measures: the Term Frequency and the Inverse Document Frequency. The former is the number of times a term appears in a document and the latter values the terms that appear in few documents. The Equation 2.4 shows how the TF-IDF is computed, where d is the document in question, t is the term being analysed, N is total number of the documents (or instances) in the collection, $tf_{d,\,t}$ is the number of times the term t is in the document d and df_t is the number of documents that contain the term t.

$$tf - idf_{d,t} = tf_{d,t} \times idf_{d,t} = tf_{d,t} \times \log \frac{N}{df_t} \tag{2.4}$$

Cosine Similarity

After having each word represented in the vector space model, similarities between them can be computed in several ways, the most common being the cosine similarity. Equation 2.5 shows how to compute the cosine similarity between the vector representation between two documents, *d1* and *d2*. The numerator is the dot product or inner product of the vectors $\vec{v}_{\mathrm{d1}}$ and $\vec{v}_{\mathrm{d2}}$, while the denominator is the product of their Euclidean lengths. The denominator normalizes the vector to unit vectors such that their lengths do not affect the result. Otherwise, two vectors could be very similar, but their similarity value would not reflect due to their length difference.

$$sim_{d1,d2} = \frac{\vec{v}_{d1} \cdot \vec{v}_{d2}}{|\vec{v}_{d1}| \cdot |\vec{v}_{d2}|} = \cos(\theta) \tag{2.5}$$

It is called cosine similarity because its value corresponds to the cosine of the angle the vectors make when represented in the same plan. Figure 2.3 shows several normalized vectors and the cosine similarity between the vectors $\vec{v}_{\mathrm{d1}}$ and $\vec{v}_{\mathrm{d2}}$ equals the cosine of the angle between them, θ.

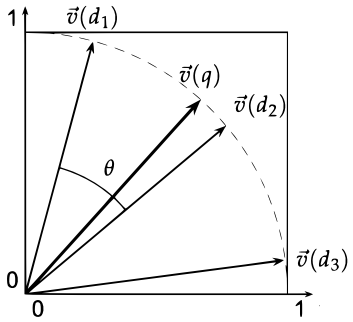

Figure 2.3: The angle used by the cosine similarity. Adapted from [23]

2.3 Entity Matching

Entity matching (EM) is the task of predicting whether or not two instances correspond to the same entity. It is widely used for data integration and cleaning given its use to identify duplicate objects [16]. More formally, having two sets of entities $A \in S_A$ and $B \in S_B$, the EM problem aims to identify all correspondences between entities in A and B that represent the same real-world object [20].

2.3.1 Predictors

Another relevant aspect of entity matching is how two entities predictors are represented for the classifier. Instead of submitting all the attributes available, some techniques relate the value of each feature using similarity measures, and those measures are the predictors instead of the original values [20]. This representation is a beneficial approach when relating two entities, given that the comparison techniques can be carefully thought out and customized for each pair of attributes.

2.3.2 Blocking Methods

Blocking is a common technique used in EM to reduce its computational cost by trying to identify which entities will likely match and thus decreasing the number of comparisons [30]. The idea behind it is to divide all the data into blocks, hence the method's name, and only compare each block's instances among themselves. There are overlapping and disjoint methods depending on the number of blocks a sample can appear [20].

2.4 Evaluation Metrics

Evaluating the classifier's performance is crucial to understand the relevance of the system for different data. For this work, it is essential to consider standard metrics, metrics for imbalanced data sets and adapt them to *multi-label* problems.

There are specific metrics for imbalanced data sets because it is necessary to consider how well the model can predict both classes and not the results in general. Otherwise, it could have an

excellent performance just by always predicting the majority class. The accuracy is not advised since there is no clear distinction between classification values on different classes.

Regarding the multi-label problem, there are specific metrics to take into account the notion of partially correct. Given that the prediction result is a set of labels, it can be fully correct, partially correct, or fully incorrect [15].

Four crucial concepts when addressing metrics are the true positives, true negatives, false positives, and false negatives. True positives (TP) and true negatives (TN) are instances that the model correctly classified as positive and negative, respectively. In contrast, false positives (FP) and false negatives (FN) are instances that the model wrongly classified as positive and negative, respectively.

For the remainder of this section, consider D to be a multi-label data set consisting of $|D|$ instances $(x_i, Y_i), i = 1..|D|$ where x_i are the predictors of the example i and Y are the set of labels associated to it. Moreover, let Z_i be the set of labels predicted by a model for the instance i.

Accuracy

Accuracy is one of the most common metrics when evaluating a classification task. It is defined as the ratio between the number of correctly classified instances and the total number of examples. Equation 2.6 shows how it can be computed.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

For multi-label problems, accuracy is the average per instance of the ratio between the correctly predicted labels and the number of all labels as shown in Equation 2.7 [15, 35].

$$Accuracy = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{2.7}$$

Precision

In a traditional classification problem, precision is the ratio between the positive instances predicted correctly (TPs) and the total number of positive instances as shown by Equation 2.8.

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

For multi-label problems, precision is the average per instance of the ratio between the correctly predicted labels, and the number of labels predicted as shown in Equation 2.9 [15, 35].

$$Precision = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|} \tag{2.9}$$

Recall

In a traditional classification problem, recall is the ratio between the positive instances predicted correctly (TPs) and the total number of instances predicted correctly. It can be computed by Equation 2.10.

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

For multi-label problems, recall is the average per instance of the ratio between the correctly predicted labels and the number of existent labels as shown in Equation 2.11 [15, 35].

$$Recall = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i|} \tag{2.11}$$

F-measure

The F-measure is the harmonic average of the precision and recall metrics. It is only high when both recall and precision are high. It is computed as shown in Equation 2.12.

$$F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.12}$$

This is a very common metric to assess the performance of models that used imbalanced data sets [13]

Exact Match Ratio

A specific measure for the multi-label problem is the exact match ratio which, as the name suggests, shows the ratio between the number of fully correct predictions and the total number of predictions. Its computation is shown in Equation 2.13 where I is the indicator function [33].

$$ExactMatchRatio = \frac{1}{|D|} \sum_{i=1}^{|D|} I(Y_i = Z_i) \tag{2.13}$$

G-Mean

Geometric mean (G-Mean) is ideal for imbalanced data sets given that indicates the tendency of two values instead of its exact value. It derives from the True Positives of each class. If at least one class is unrecognized by the classifier, G-mean resolves to zero.

AUCRoc

AUCRoc curves are performance measurements used for classification problems that attempt to describe a model's ability to distinguish between classes. ROC is a plot that relates True Positive Rate with the False Positive Rate. The Area Under the Curve (AUC) is one value to assess the model's performance that corresponds to area below the ROC chart. It can vary between 0 and 1.

Chapter 3

A review of methods for matching tasks and people

Human resource allocation has been studied thoroughly using several methods and in different applications areas such as health, production planning, tourism and hotel services, project management, maintenance management, among others [5]. However, many of these studies focus on optimizing the allocation by maximizing or minimizing a measure (profits, quality and time, costs, respectively). As previously discussed, that is not this dissertation's aim, given that the tasks to be allocated are not identical and neither are the people available. Hence, the goal is not to distribute all the tasks. It is to find the perfect match between a task and a person. Due to the difference between most studies done in the HRAP (Human Resource Allocation Problem) and the current problem, optimization studies were not considered for the present literature review.

There was a focus on the open-source and crowdsourced development domains. There have been attempts to automatize the assignment of a job to the most appropriate worker or at least to triage the best candidates. These contexts have a great variety of developers available and, therefore a more significant hardship in finding the best fit for a task. Therefore, these domains are more similar to the problem in question and were carefully studied.

The present chapter systematizes the Literature Review on acquiring the most appropriate person for a given task. It is divided into two sections. The first one regards studies applied to the crowdsourcing domain, and the second one describes works in the open-source domain. For each work, it is presented the main objectives, the data used, the approaches tested, and which provided the best results.

3.1 Crowd sourcing tasks

In order to better understand the works in this section, a brief explanation of how crowdsourcing platforms are necessary. In these platforms, some entities (individual people or organizations) can submit a task with its description, due date, and some might have a compensation [6]. Anyone can submit a solution for any task on the platform. The best one (called the winner in some of the

works) is selected by the task owners. Sometimes, several other solutions are highlighted to give a positive evaluation to other people and for them to have a good track record in the platform. In this domain, there are many tasks for people to choose between [9] and a considerable amount of solutions from which the winner must be selected.

Yang et al. (2016) [41] recommends tasks to users in the context of crowd-sourced software development in order to diminish the number of developers that sign up to a project but do not submit a solution. For each task, it computes a set of similar tasks with a distance function computed from the difference of some of the tasks' attributes such as prize, registration and submission date, type, technologies, among others. Features are computed from the similarity subset of tasks that better reflect how suitable a given user is to perform that assignment. Finally, for each day, a Random Forest classifier was developed to predict if the user would be a submitter, a quitter, or the winner. The tasks were ranked using the probability of the user being the winner of the task. The model reached precision values of 99% for the quitter class and 78% for the winner class. No other models were considered. Therefore, no conclusions can be drawn regarding the most appropriate classifier for the task.

Mao et al. (2015) [24] presents two content-based recommender systems to ease the task selection process. One suggests participants that could win the given assignment, whereas the other showed people that could be interested in working in it. The predictions were based on developers' previous activities and only considered complete data and tasks with the same domain as the designated task. Moreover, for the potential winners' recommender, the developers with less than five wins were discarded. The attributes used were numerical and textual - date, programming language, title, description, duration, and payment. Textual tasks are transformed into a word vector using NLP techniques. The labels were the winners of the task and the participants, which made one of the recommenders a multi-class, single-label problem and the other a multi-class, multi-label one. Afterward, several classifiers were trained and applied to the data, namely, C4.5, Decision Tree, Naive Bayes, k-Nearest Neighbour. The multi-class, multi-label problem was transformed into a single-label problem by combining every task with every participant. Finally, based on the probability distribution generated by the classifiers, a ranking with the top N developers was created. The datasets used were extracted from TopCoder, and experiments were made to recommend the top 5, 10, and 20 developers. Mao et al. (2015) used the hit rate, average precision, and diversity as metrics. The former showed the percentage of recommended winners considering all the tasks. The average precision indicated the mean for each assignment of the percentage of correctly suggested participants. Finally, the diversity measured the number of developers that the system could recommend. When suggesting potential winners, the results were better than the considered baseline for the C4.5 and Naive Bayes classifiers by 12% and 7%, respectively. In regards to the other recommender, the average precision improved only with the C4.5. classifier by 6%.

3.2 Open source Development

As it is widely known, open-source development mainly relies on volunteers to fix bugs or provide new features. However, there must be a group of people who manage the changes and assure their quality. They usually know more about the project and function like a work team. The present studies focus on assigning those people tasks such as review a pull request, among others.

Anvik et al. (2006) [2] uses a machine learning approach to suggest a list of the most fitting developers to solve the bug. These recommendations are based on the work previously done by the developers, meaning based on other bugs that were assigned to them or bugs they ended up fixing. It is a semi-automated approach given that the final decision of which developer should be responsible is made manually. The chosen methods were based on text categorization, given the free-text nature of a bug report. Therefore, the dataset used is constructed with features extracted from the bug reports using natural language processing pre-processing steps and the developer's name as the label. Some of the bug reports were not considered because the label was unknown, the developer was not available anymore, or they had not been responsible for at least nine bug reports in the last three months. Support Vector Machines was the best algorithm compared to Naıve Bayes and C4.5 and reached precision levels of 57% and 64% on the Eclipse and Firefox development projects, respectively. However, when validating with the gcc project's data, the precision was down to 6%, which the authors attribute to several reasons: the imbalanced distribution of bug reports between developers and faults in the labeling process.

Yu et al. (2016) [43] adapts the work done regarding the recommendation of developers for code review and bug assignment to a similar domain, the review of pull requests. It tries approaches using three different techniques (Machine Learning, Information Retrieval, and File Location) and a new one that recommends the developers based on a Comment Network that it constructs. Its title and description describe a pull request. The labels correspond to all the developers that commented on it. The data preprocessing steps include removing stop words and non-alphabetic tokens, stemming the remaining words, and converting the text to a weighted vector representation. Each element of the vector is the value of a term computed using TF-IDF.

Regarding the machine learning approach, only the SVM model is used to predict the probability of a developer commenting on the pull request. The developers are ranked according to this probability. Given that several developers can comment the same pull request, it is a multi-label classification problem and it uses the one-against-all strategy. For the IR based approach, the cosine similarity between a new pull request and all others is computed. Then, the developers are ranked by their expertise, represented by the sum of the similarity of the pull requests they have reviewed. In the File Location based approach, this expertise is computed based on the path similarity of the files of the reviewed pull requests with the current one. The similarity of two files is the ratio between the number of equal components of the file path and the maximum number of components. Furthermore, the similarity of two pull requests is the sum of the similarities of every file in the two pull requests. Finally, the intuition between the comment network is that developers with similar interests to the pull request submitter will be better reviewers. Therefore,

for each project, a graph is computed. The nodes represent the developers, and the edges are the interactions between users. Their weights depend on the number of comments in pull requests, the number of pull requests in which there are comments, and the comments' age, favoring more recent ones. In order to recommend the most appropriate developers, Yu et al. uses Breadth-First Search. When the pull request in question is the first one to be submitted by that developer, the algorithm is not viable, and therefore another strategy is implemented. It is always assumed that the developers have reviewed other projects, and they are not entirely new to the community. Hence, it is possible to create groups of users that share the same interests. In practice, this means users who comment and interact with each other in the same pull requests belong to the same group. The users in the community of the submitter are recommended. For all experiences, the precision, recall, and f-measure were computed for different numbers of recommended developers. The best approach is concluded to be the IR-based one with an F-Measure of 50.05%. All approaches achieve the best performance when recommending 4 or 5 developers. Hybrid approaches are also tried and can achieve significant improvement.

Xia et al. (2015) [39, 40] proposes the DevRec method in order to automatically recommend a list of developers that are likely to solve a given bug report. DevRec is a mixed method that performs two kinds of analysis, one based on bug reports, other based on developers. The former finds the k-nearest neighbors to a given bug report. Based on the solvers of the neighbors, it computes a score for each developer. A vector with four features characterizes each bug report: the terms (the number of times every stemmed word of the document appears), the topics (a vector with the probabilities of the report to be about a specific topic, computed using latent Dirichlet allocation), the product (a vector of binary features that are set to one if the bug in question will affect the matching product), and the component (also a set of binary features each of them equal to one only if the bug will influence the correspondent component). The distance of two bug reports equals the Euclidean distance of the vectors that represent them. Afterward, the probability of each developer be assigned to the bug report is computed using a multi-label classification model - the ML-KNN approach. The developer based analysis finds the distance between people and reports instead of between two reports. The same features are used but are computed so that the developer's experience (meaning all the bug reports one might have solved) are considered. Finally, both analyses are summed, assigning different weights to each feature. The best value for these weights is found using a subset of all the bug reports. The approach is compared to Bugzie and DREX using two well-known information retrieval evaluation measures: recall@5 and recall@10. They vary from 48.26% to 79.89% and from 60.63% to 89.24%.

Wu et al. (2011) [38] brings DREX, a developer recommendation for bug resolution using KNN and expertise ranking. It starts by getting a set of the most similar bug reports. This similarity is computed using the cosine similarity between the two vectors representing each bug report. Afterward, all the developers that participated in the bug resolution or commented in at least one of the K most similar reports are extracted. These are the people who are ranked and recommended to solve the bug report. Seven different criteria are considered for ranking them: one based on the number of similar bug reports they interacted with, the rest using social network analysis

(Indegree, Outdegree, Degree, PageRank, Betweenness, and Closeness). The social network's nodes are the developers and its edges are the sum of the cosine similarity between the new bug report and the bug reports with which both developers have interacted. Precision and Recall were computed to evaluate the method's performance. The best performance was obtained using the Out-Degree and Frequency metrics and had a recall of 0.6 on average when recommending the top 10 developers.

Naguib et al. (2013) [28] proposes a new method for recommending people to fix bug reports using the former's activity profile. The first step is to discard stop words, HTML tags, hexadecimal numbers, and numerical information and process the rest of the content by a lemmatizer to group similar words with different inflections. Afterward, the LDA model uses the processed terms to compute a matrix that relates each bug report to each topic, in which a value is the number of times a topic is associated with a report. The matrix is normalized in a way that all values are less than one. Thus, it is possible to compare the values of different reports. Then, a similar matrix is computed to correspond the developers to topics. Each matrix's row corresponds to a developer and each column to a topic. The values show how much the users were related to a topic in previous bug reports. Finally, the developers are ranked according to the users' expertise on that topic and the roles (reviewer, assigner, and resolver) they had on previous bug reports. The evaluation metrics used by Naguib were Actual Assignee Hit Ratio (meaning if the list of recommended assignees contains the actual assignee of the new bug report) and Top-n Hit (top-1, top-3, top-5, top-10). The described approach can achieve an average hit ratio of 88%.

Yu et al. (2014) [42] extends the traditional Machine Learning approach for assigning bug reports and applies it for the pull request review. Furthermore, the same work proposes a new method for the recommendation based on the comment networks. The machine learning approach considers the text describing the pull request and uses a vector space model to represent it. Each value in the vector is the importance of a term in the pull request computed using the TF-IDF measure. Only the relevant terms are considered, meaning, stop words, non-alphabetic tokens are removed, and the remaining words are stemmed. The SVM model is used with the one-against-all strategy. Afterward, the developers are ranked according to the probability the classifiers predicted on the label. The comment network approach starts by constructing the network for each project. It is defined as a weighted graph where the developers are the vertexes, and there is one edge for each pull request two people share. The weights are influenced by how close in time the comments of both developers are. The recommendation uses two techniques depending on whether or not it is the first pull request of the user in question. For users with history, developers that have previously interacted with the pull request submitter are recommended. Hence, an adapted Breath-First Search is used to obtain the top-k developers. For users who have not submitted a pull request but have contributed or commented, clusters of developers are created, and those belonging to the same cluster or close ones are chosen. Recall and precision are the metrics used to prove the pertinence of the method. The results show that the CN-based approach reaches a precision of 78% and 67% for top-1 and top-2 recommendations, respectively. The ML-based approach reaches 73% precision of top-1 recommendation, and both methods get to a recall of 77% for

top-10 recommendation.

Lee et al. (2017) [21] uses deep learning for the bug triage problem. It starts by adopting Word2Vec to convert each word to a vector form, which appropriately handles multi-lingual bug reports and jargon in those same reports. A matrix is created in which each row corresponds to a word's representation using Word2Vec. The attributes considered when computing this matrix are the report's summary and description. This matrix is the input for the CNN model that computes which developers are to be assigned to the report. The results are evaluated using Top-K accuracy and are compared to the results of Human Triagers. The top-1 accuracy improvements for four of the projects tested are 19.44%, 10.73%, 18.28%, and 12.55%, respectively. However, for two other projects, the performance improvements are less than 3%.

3.3 Overview

From the present literature review, one can extract important conclusions. Firstly, the studies show the importance of pre-processing unstructured text using tokenization, stop word removal, lemmatization, and stemming. Furthermore, SVM is the most used classifier for these tasks and has proven its success. It outputs the probability a person has to be allocated to a task and ranks the people by that value. Moreover, some studies handle the problem using a multi-label approach where the task's attributes are the predictors and the developers the labels. Some even choose to relate all tasks with all people and rely on binary classification. Finally, the most used metrics are recall and precision.

Chapter 4

Problem description and Proposed Approach

The present section defines the problem to allow it to be tackled as a multi-label problem. Furthermore, the proposed approach is introduced with details of each step and justifications of their importance. Finally, the data set used to validate this approach is described given the known impact of data understanding in the methods used to handle it [37].

4.1 Problem

As stated in Chapter 1, this dissertation aims to be able to recommend the appropriate people for that job when given a task. Some parameters describe the task and all the people available to makes this recommendation possible. Human resource allocation is an essential process in every company and a difficult problem to be solved manually, given the amount of knowledge a person must be aware of to make a good decision. Since machine learning has been proven successful in other domains in systematizing data and extracting patterns and conclusions from data that are not so obvious when analyzing manually.

In order to apply machine learning, namely classification techniques, to a problem, it is essential to first define it as such. In other words, one has to decide which information should act as predictors and which should result from the framework. For this problem, as in works present in the literature review, the apparent predictors are the attributes that describe a task, and the output should be a set of people that are a good match for the job. Hence, this problem is a multi-label one. A relevant question is how to include the data regarding the people and how pertinent it is to the framework's success.

4.2 Data Understanding

As mentioned in Chapter 1, the present work inserts itself in the PROMESSA project and aims at providing help to project managers when allocating human resources to tasks. Hence, the data used to study and validate this methodology is expected to be taken from that same project. The data was extracted from the project management software used by Fraunhofer [1] (Project Control) and stored on a MySQL database. This section presents the extracted data, namely, its structure and content.

The database is made up of 35 database tables which include 1126 different projects starting between 2013 and 2019. The database tables related to the problem's domain are Project, Baseline, Work Package, Task, User, Allocation, Keywords, Categories, and other tables that relate some of these. Figure 4.1 shows the UML class diagram of the models that were considered.

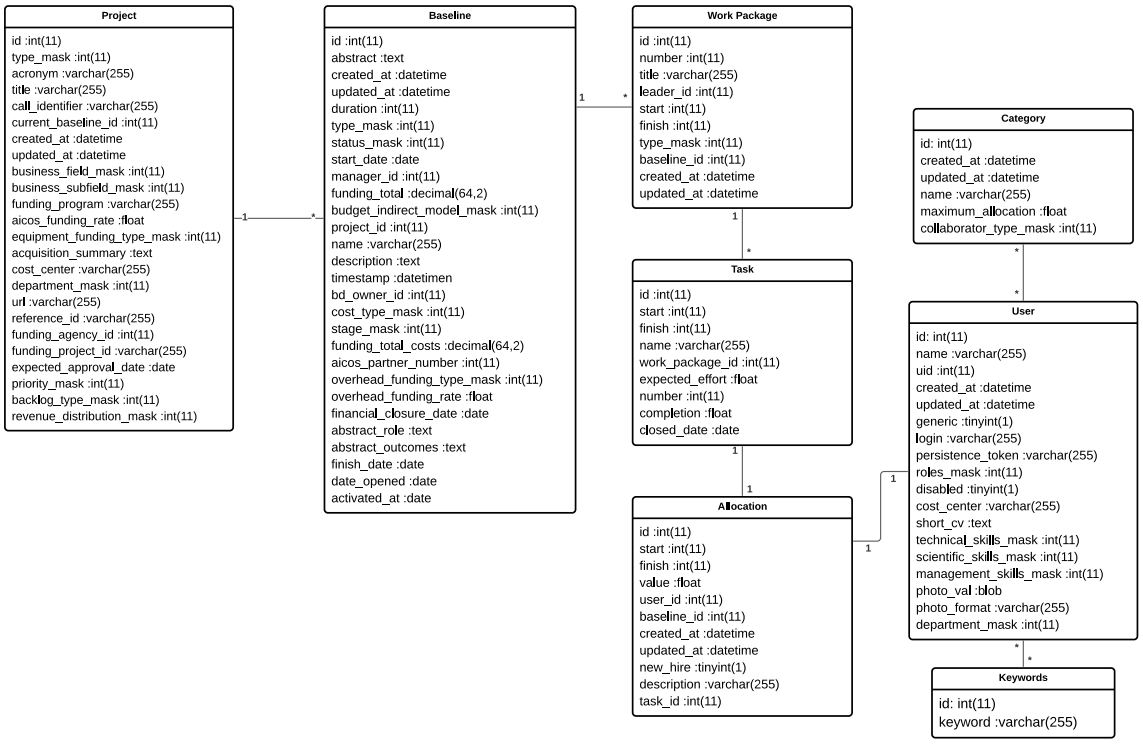

Figure 4.1: UML of the considered database models

The `Project` table corresponds to all the information related to a project such as the business field and subfield, the dates, the funding program and details and among others. As mentioned the projects are all created between 2013 and 2019, being 2013 the year with the most projects (278) and 2014 the one with the less projects (111). There is a lot of attributes characterizing a project: some of them about the project itself (*acronym*, *url*, *reference_id*, *type_mask*, *priority_mask*, *backlog_type_mask*, *call_identifier*), some describe how the project came to be (*acquisition_summary*), some regard the area it is inserted in (*department_mask*, *business_field_mask*, *business_subfield_mask*), some regard the funding and the revenue of the program (*funding_program*,

[1] https://www.aicos.fraunhofer.pt/en/home.html

funding_agency_id, *funding_project_id*, *aicos_funding_rate*, *equipment_funding_type_mask*, *revenue_distribution_mask*, *expected_approval_date*, *cost_center*), some support the database organization (*id*, *current_baseline_id*, *created_at*, *updated_at*), some don't reflect real or understandable information (*title*). The attributes considered relevant for this work are *type_mask*, *department_mask*, *business_field_mask*, *business_subfield_mask*. The *current_baseline_id* is also helpful, not to match the tasks to the users but to later on link a baseline to its project. The *type_mask* attribute distinguishes the projects between European, National, Industry, Internal, Reserved, a Thesis or an Other type of project. The *department_mask* can be one of eight departments - research and development, databases, financial, governmental, human resources, IT or administration and legal. The business fields can be one of three values (Ambient Assisted Living, Information and Communication Technologies for Development, Non-Research and Development and Others).

The `Baseline` table represents an instance of a project in a moment in time and can be defined in this context as a project checkpoint. A new baseline is created when a project is redefined or the deadlines change and hence the most relevant one is usually the most recent one. In the baseline is stored the data in which a project starts, how long it will take, the manager, among others. There are a total of 6 267 baselines of 1 126 projects (so there is at least one baseline for each project) with the start date varying between 2010 and 2021 (which means some projects were added to the database after they were created) and the finish date varying from 2012 and 2025. As the Project table, the Baseline also has a lot of attributes characterizing it: some of them describe the baseline itself (*type_mask*), some regard the financial aspects of the baseline (*funding_total*, *budget_indirect_model_mask*, *funding_total_costs*, *aicos_partner_number*, *overhead_funding_type_mask*, *overhead_funding_rate*, *cost_type_mask*, *financial_closure_date*), some show the person responsible for it (*manager_id*), some store the baseline timeline and status (*start_date*, *finish_date*, *duration*, *status_mask*, *stage_mask*, *date_opened*, *activated_at*), some support the database organization (*id*, *created_at*, *updated_at*, *timestamp*, *project_id*) and some don't reflect real or understandable information (*abstract*, *abstract_roles*, *abstract_outcomes*, *name*, *description*).

The `Work Package` table represents the subdivision of the project tasks in almost a themed way. It is characterized by a *title*, a *start* and *finish* numbers, the baseline it corresponds (*baseline_id*), a *type_mask* (which can correspond to research and development, demonstration, management, and other), a *number* to represent the order in which it was created. The *start* and *finish* numbers are the months after the start of the project (which is saved in the baseline) in which this work package will start and finish, respectively. There is also the *leader_id* attribute that indicates which institution is responsible for the work package. There are a total of different 22 462 work packages in the database linked to 5 149 different baselines.

The `User` table has the details of every worker in the organization. It has information about their skills (*technical_skills_mask*, *scientific_skills_mask*, *management_skills_mask*), their role and place in the institution (*roles_mask*, *department_mask*), other information useful for the company (*disabled*, *uid*, *generic*, *cost_center*), attributes that don't reflect real or understandable

information (*name*, *login*, *persistence_token*, *short_cv*, *photo_val*, *password_hash*) and finally some that support the database organization (*id*, *created_at*, *updated_at*). The most important attributes are *roles_mask*, *department_mask*, *technical_skills_mask*, *scientific_skills_mask*, *management_skills_mask* whose values are detailed below.

- **Role** - admin; collaborator; human_resources; group_leader; marketing_communication; administrative; director; project_mgmt_office; head_of_department; business_developer; finance
- **Department** - New Business Development, Governance, Administrative, Research and Development, IT, Finance, Human Resources, Legal, Marketing and Communication
- **Technical Skills** - Android Dev (Java, XML, JSON); Linux Dev (C, C++, Python, Perl, Shell Scripting); IoS Dev (Objective C); Windows Dev (C#, .NET); Web Dev (PHP, Ruby on Rails, HTML 5, MySQL); HW Dev; SW Testing (Specifications, Tools, Methodologies); Databases (SQL, PL-SQL)
- **Scientific Skills** - Image Processing; Signal Processing; Sensor Fusion; HCI and UX; Natural Language Processing; Data Mining ; Networking and Telecommunications; Industrial Design, Design (incl. Motion Design); Cloud Computing
- **Management Skills** - Project Management skills (Scope, Cost, Time, Risk, Quality, Procurement, Communications and Team Mgm.); EU Proj. Management (as project coordinator); QREN Project Management (as project coordinator)

The users can also be described with their link to another table - `Keywords`. It can take 102 different values and one user can be linked to more than one keyword and these values are related to the user areas of specialty. Some examples of the keywords' values are embedded dev, digital systems dev, biomaterials, bioengineering, software architecture, among others.

The same way, each user is associated with an instance from the `Category` table depending on the job position. A user can have several categories and a category can be one among 61 values such as Service Provider, Research Grant (BI) - Doctor, Research Grant (BI) - Bachelor, Group Leader (FhP_Sup_26), among others.

The `Task` table includes all the information of a given task such as its *start* and *finish*, the *name*, the work package it is included in (*work_package_id*) and the order in which it was created (*number*). Furthermore, outside of the scope of this work, 12% (8 271) tasks have been classified in a way they have been assigned some of a set of 24 labels according to the subject they cover. Another important aspect is that there are instances with the *name* attribute in Portuguese and others with it in English. There are a total of 69 185 tasks in the database, linked to 21 568 work packages.

The `Allocation` table relates a user (*user_id*) to a task (*task_id*) and saves information such as how much time that task will take from the user (the *value* is the percentage of the total time the

task will occupy), the baseline it concerns (*baseline_id*), a description (*description*) and if it was needed to hire a new person for that task (*new_hire*).

4.3 Data Construction

One of the challenges of this problem is how to relate and represent all the information available from the two entities - people and tasks - in a manner that helps the model predict whether or not they are a match. Two techniques were used.

The first one only considers the history of a person based on the intuition that the people should be allocated to tasks that are similar to the ones they have performed before. Therefore, the classifier will rely on the jobs that employee has performed for the company. In this case, none of the information regarding the person is considered so only the tasks' data are used as predictors. For each person, a model is tested and validated independently to predict whether or not that person is appropriate to each task.

The second technique corresponds to the Cartesian product between all the tasks' and people's data. The classifier's performance will depend on its ability to establish a correlation between the data that describes the task and the one that characterizes each person. Only one classifier is needed but the number of instances and predictors are significantly greater.

4.4 Entity Matching Techniques

The present work is somewhat similar to entity matching problems. Hence, some of its techniques and concepts are integrated into this approach.

Entity Matching problems recur to the ratio or difference between attributes that represent identical knowledge. For example, consider two data sets of movies where the goal is to find the instances in the two sets that correspond to the same real-world entity, namely the same movie. Both sets will probably have an attribute with the movie's title. A typical entity matching technique considers the difference or similarity between the two titles instead of the two attributes separately. This way, the classifier is provided with much more concise and pertinent information.

However, for the problem at hand, the two entities to match - people and tasks - do not correspond to the same real-world object. The attributes that describe them cannot be directly compared, given that they characterize the entities in very different ways. This idea can be adapted to the present work by computing the similarity between the two texts that describe a task and a person. Based on the intuition that the exact keywords will be used to depict a task and the skills of the person allocated to it, the similarity between the texts could be a valuable predictor. Hence, a new feature that represents the cosine similarity between the two entities' texts is taken into account.

Furthermore, entity matching data sets are highly imbalanced because the number of instances to compare corresponds to the Cartesian product of two different sets where only a reduced fraction represents the same entity. Some frameworks recur to the so-called blocking methods that help

reduce the number of needed comparisons. They divide the entities into sets trying to aggregate instances with the same characteristics. The goal is to compare entities that belong to the same group, reducing the number of negative instances submitted to the classifier. This technique can be done manually, using restrictions demanded by the business domain, or automatically, using clusters techniques to form those groups. A restriction based on the current domain is imposed in some experiments, and a correlation between a person's and a task's department is used.

4.5 Text Classification Techniques

As seen in Chapter 3, most of the studies present in the literature review handle textual attributes such as bug reports' reports, pull requests' descriptions, among others. In this data set, there is no unstructured text that describes the instances. However, all the relevant attributes can be converted to strings. Therefore, it is possible to create a text that characterizes each entity without losing any vital information. Afterward, the same text categorization methods can be applied.

Chapter 5

Experimentation

The present section describes in more detail the required steps to make the experiments possible, their relevancy to the final result, and the conclusions that can be extracted from each one of them. The data used is detailed as well as the reasons for discarding some information. The pre-processing and data construction techniques are also described.

5.1 Data used

With a better understanding of the data and the project in hand, one concludes that not all the information is relevant and should be considered to validate the problem in question. The present section shows which data is discarded and explains the reason behind each filtering.

Only the most recent Baselines were taken into account, so there is no duplicated or outdated information. Furthermore, from each baseline, the only field used is the *type_mask* in order to make that filtering possible.

The work packages fields considered are the *baseline_id*, in order to link tasks to baselines and to filter out the work packages that do not belong to the most recent baseline and the *title* given it describes the work package and hence the theme to which a task belongs.

A person's most important attributes are *roles_mask*, *department_mask*, *technical_skills_mask*, *scientific_skills_mask* and *management_skills_mask*. There are 432 different people in the database, but only those with more than 5 tasks were considered. There was an attempt to narrow the possible people for each task even more based on the intuition that not all people are always available and can leave or enter the company. The *created_at* attribute should depict the date from when a new person enters and the *disabled* along with the *updated_at* should portray the date until when a person is in the company given that the information is updated when the person leaves the company. However, when comparing these dates to tasks the people are allocated to, there is a clear mismatch because people are allocated to tasks that started before they entered the company and tasks that finished after a person left the company.

Regarding the tasks, some are discarded because they do not have a valid person allocated to them or do not belong to the most recent baseline. Only a task's *name* and the *work_package_id* it is related to are deemed relevant.

From the allocations table, the *description* attribute is not taken into account because it is inserted after a person is allocated; hence the approach should be able to recommend without it. There are 14 8261 tasks, 0.19% (282) are an exact match of an existent allocation, 0.08% (117) only were created or updated at a different time, 0.07% (102) has a different description, 0.43% (644) only differ on the time it will take up. Moreover, only 329 people and 54 153 tasks are allocated from 5 049 different baselines.

Table 5.1 depicts for each model the actual number of instances, and how many of those are considered: all the projects, the most recent baselines, the work packages linked to the most recent baselines, the people allocated to more than five tasks, the categories, and keywords related to valid people, the allocated tasks that belong to a valid baseline and the allocations that relate valid tasks and people.

Table Name	**All Instances**	**Valid Instances**
Project	1 126	100% (1 126)
Baseline	6 267	18% (1 126)
Work Package	2 2462	15% (3 455)
User	432	45% (194)
Categories	61	44% (27)
Keywords	102	87% (89)
Task	69 185	11% (7 565)
Allocation	148 261	9% (12 772)

Table 5.1: Number of filtered instances of each table

5.2 Data Preparation

Unlike most of the work reviewed, the available database does not have a report or other text that describes neither the person nor the task. Hence the methods must be adapted. The attributes that better describe the person, tasks, and projects are strings but stored in different formats - some are just strings, others are masked or encoded - and some are even in different languages. Furthermore, there are several ways to represent strings attributes that can improve the results of a Machine Learning model.

Table 5.2 details the existent attributes, which of them describe the people and the tasks and the format and language used to store them in the database.

	Instance	Format
Person	technical_skills_mask	Encoded
	management_skills_mask	Encoded
	scientific_skills_mask	Encoded
	roles_mask	Encoded
	user_department_mask	Encoded
	keyword	String
	category	String
Task	project_type_mask	Encoded
	project_department_mask	Encoded
	business_field_mask	Encoded
	business_subfield_mask	Encoded
	work_package_title	String
	work_package_type_mask	Encoded
	task_name	String
	labels	String

Table 5.2: Format of the string attributes used

Handling Multiple Languages

The present section details the steps needed to handle attributes in different languages and formats and to compute and correctly represent a textual parameter.

In order to tackle the problem of having a data set with multiple languages, it is essential to start by understanding the number of instances in each language. The library Langdetect[1] is used to extract the language of each task's name and classifies 57% (4 319) of the tasks as English, 15%(1 158) as Portuguese, and the rest 28% (2 088) as other idioms. This classification is not 100% accurate given it says the dataset includes many different languages, and we know that is not true. However, it is difficult to extract an idiom from concise sentences, especially since many Latin languages have very similar words. For this study, it is considered that there are 57% tasks stored in the English language and 43% stored in Portuguese.

There are different possibilities for handling the different languages on the dataset - one can discard all the instances of one of the languages, design an approach that does not depend on it, and keep all the original values or translate some values to standardize the data. It would only make sense to discard the Portuguese values because the remaining attributes are all in English. However, removing them would reduce the data set to almost half. That would be a significant loss of data which would make the work less reliable and valuable.

[1] https://pypi.org/project/langdetect/

Regarding the second possibility, the literature review shows a significant dependency on language to remove stop words and stem the remaining content, making it hard to put into practice. Therefore, the most appropriate approach for this case is to translate the Portuguese values into English. Given that only the tasks' names have to be translated and they are not very complex texts (although the longest one has 26 words, on average, they have 4.6 words), the error that might come from this operation is not relevant.

The python library Deep_translator[2] is used to translate the task names that are in Portuguese. Given the restrictions on the number of requests per second to the Google server, it is still a relatively time-consuming operation to translate 3 246 sentences. There must be an interval of 2 seconds between each request, and the processing takes up to 2 hours and 20 minutes.

Standardizing the Data

In order to check which format led to the best results, three data formats are considered. For some experiences, the attributes are encoded, and for others, they are all represented by their original string value.

The person's keywords and categories are concatenated given that some people had more than one keyword or category and then encoded, resulting in 37 and 98 different keywords and categories, respectively. The same approach is applied to the task's labels deriving 180 different labels. Finally, the work packages' title and the tasks' name result in 1849 and 5888 unique values, respectively.

Regarding the conversion from masked values to strings, in most attributes (like the project's type, department, and business fields), it is straight forward being only necessary to replace the integer by the string in represents. However, for a person's technical, management, and scientific skills, the transformation is a little more complex, given that a person might have more than one skill for each area. For these cases, a skill is represented by a bit, so it is necessary to extract them differently. In the end, there are 43, 5, and 34 different combinations for technical, management, and scientific skills, respectively.

NLP pipeline

With all attributes converted to strings, it is now possible to process them as any data in the NLP domain. Firstly, all features are concatenated in one unique text to represent that instance, resulting in an average of 145 words per text for tasks and 248 for people. Afterward, all the numeric and non-alphabetic symbols are removed, and the text is converted to lower case. Then, the stop words are discarded, given that they do not add any value to the text. Finally, the words are stemmed using the Porter Stemmer, and the texts are composed of 76 words for tasks and 123 for people. This processing is done using the NLTK library. The processing reduces the number of unique words from 3880 to 1574 and from 266 to 138 for the tasks' and people's texts, respectively. The

[2] https://deep-translator.readthedocs.io/en/latest/

(a) Tasks' texts wordcloud

(b) People's texts wordcloud

(c) Tasks and People's texts wordcloud

Figure 5.1: Wordclouds extracted from the tasks' and people's texts

number of unique words corresponds to the number of features in BoW representation. Hence, these reductions significantly impact the models' accuracy and efficiency.

Figure 5.1 shows the most recurring words in the texts that categorize the people and the tasks. There are not many words common in both charts, only standing out develop, research and project. However, the word research that mainly comes from the people text seems to be even more frequent than words such as *develop* and *project* that appear a considerable amount of times in both. There seems to be a more sparsity of common words in the tasks' text than in the people text, which justifies the greater number of words common to both 5.1b and 5.1c.

5.3 Train and Test split

The instances are split into two sets in a 70%-30% proportion to validate the experiments. For this division, some restrictions must be followed to emulate the actual behavior of the framework in the best way possible. Hence, two restrictions are taken into account.

The first constraint considers the time dependency of the data. In other words, it is not reasonable to validate a model with events that happened before the ones with which the model was trained. If used in the everyday life of a company, the framework would never have access to future information to make predictions about current events. Therefore, the tasks are ordered by creation date before they are split so that the instances that validate the models are the most recent ones.

The second restriction regards which tasks are allocated simultaneously by a Project Manager and hence must be in the same train or test set. When managing a project, it is common to have a group of tasks to distribute between the workers and not just one task at a time. For the framework,

these sets of tasks will be handed out at the same time. Therefore, as in the previous constraint, it is not logical for the platform to predict tasks based on information it would not have access to in the real world. For the current data set, it is assumed that tasks that belong to the same work package are distributed simultaneously. Consequently, when splitting the tasks instances in two, tasks of the same work package must be in the same train or test group.

Imposing these restrictions makes the training and testing groups include 5298 tasks (71%) and 2267 (29%), respectively. It also brings some problems regarding the distribution of positive and negative instances in both sets - 28 people had to be excluded because they only had positive instances in one of the sets.

5.4 Text-based Similarity Predictor

The text-based similarity between a task and a person needs their attributes to be represented as a vector. Therefore, they must go through all the steps described in the previous sections: their language and format must be standardized, all the attributes joined, the stop words and numeric characters removed, and the remaining words stemmed. As a result, each person and task has a text to characterize it, making it possible to represent these texts as vectors using BoW or TF-IDF.

With these representations, the predictor can be computed using the cosine similarity measure.

5.5 Blocking Method

As mentioned previously, blocking methods are used in Entity Matching to reduce the number of comparisons to be made and thus the data set imbalance. In the present data set, each task and each person is described by a department. A one-to-one correlation cannot be done in all departments, and there are missing values from some instances. However, the technique eliminates 99 827 instances.

5.6 Evaluation Metrics and Methods

Given the problem transformation approach taken to handle this multi-label problem, the work can be evaluated using two different methods. Both of them are highly relevant to prove the work's validity and look for improvements in it. Hence both are considered and systematized.

The first one regards the classifier's performance itself, meaning how well specific attributes, models, and formats can predict whether or not a person should be allocated to a task or a task and a person match, depending on which data construction method is being used. With this approach, it is easier to understand the data imbalance between positive and negative classes. Table 5.3 shows the percentiles of the number of positive samples per person. There are 5298, 2267, 7565 tasks in the train, test, and complete set, respectively. Regarding the data construction that uses the Cartesian product of people and tasks, there are 12772 positive instances and 1454838 negative instances.

Percentile	All	Train	Test
100%	618	464	154
75%	99	79	32
50%	45	28	4
25%	14	10	0
0%	6	1	0

Table 5.3: Positive number of tasks instances per person percentiles

As can be concluded, the classifiers will have to work with a highly imbalanced data set. This demands specific metrics. If only the accuracy is measured, the classifier will do very well, even if it always predicts that a person should not work on a task. It will be right 93 % of the time. Therefore, F1 Score, the AUC, and the Geometric mean are computed.

The second method evaluates the more general problem. Namely, it considers the number of correctly evaluated people and how they place in the ranking generated by the classifier. This method evaluates the framework as a whole and its capacity to predict the correct labels for the multi-label problem. Hence, measures like Exact Match, Accuracy, Recall, F1 Score are used. Another interesting metric is introduced, which depicts the number of people that the person needs to show to include all the people allocated to a task.

Another aspect to consider, especially when evaluating the data construction method based on person's history, is that there are several classifiers in question - one for each person. Hence, the evaluation metrics must reflect the performance of the set of classifiers. With this intent, the person with the most positive instances (referred to as Person A in Chapter 5) and two means are considered. The first one is the average considering all the classifiers, and the second one (Top Mean) only considers people with more than 100 positive instances in the test set. Table 5.4 shows the number of positive and negative instances in the train and test sets for each person and each mean.

	All	Train	Test
Person A	618	464	154
Mean	74.7	52.3	22.3
Top Mean	331.2	188.5	142.7

Table 5.4: Positive number of tasks instances for person A and averages

5.7 Chosen Classifiers

The approaches are tested with two classifiers: SVM and Naive Bayes. SVM was the most used classifier in the literature review, and it is prevalent in text classification problems. Therefore, it

is an excellent model to be used for this problem. Naive Bayes is also a suitable classifier for text classification and has the advantage of being more efficient to its simplicity. For example, it was only possible to compute the Cartesian Product with the Naive Bayes classifier.

Chapter 6

Results and Discussion

The present section shows the achieved results for each of the previously stated approaches and the established baseline. Both evaluation methods are shown. Different combinations of the described approaches are computed to conclude with more confidence which ones should be used. There are results with the two data construction methods (person's history and Cartesian product), the three different formats of data (masked, BoW and TF-IDF representation), the two methods of blocking (using person's department and cluster), the two models (SVM and NB), the inclusion of the text-based similarity predictor. This section only presents some results and metrics, but the complete information regarding the experiences can be found in Appendix A.

6.1 Establishing a baseline

As seen in Chapter 3, a lot of Information Retrieval (IR) techniques are used when matching people and tasks described by texts. A general approach is to compute for each person the similarity between the current task's description and the ones previously handled by a person. Then, a rank can be computed with the people that contacted the most similar tasks at the top. The described approach is straightforward but typical in recommender systems [25]. Furthermore, it helps prove the stated hypothesis given that it is not a method that recurs to Machine Learning. Unlike the studies in the literature review, it is possible to compute the similarity between each task's descriptions and each person's attributes with the data provided.

Table 6.1 presents the results obtained from computing the cosine similarity between tasks and people. A positive instance corresponds to all the values that are equal to or greater than 0.5. Table 6.2 shows the evaluation metrics for the problem as a whole, meaning, as a multi-label problem given the people are ranked by their cosine similarity to a task.

These results will serve as a baseline for the remaining sections.

	F1 Score	**AUC**	**Geo Mean**
Person A	0.000	0.500	0.000
Mean	0.003	0.276	0.292
Top Mean	0.000	0.500	0.000

Table 6.1: Baseline results of the individual predictions with BoW format

Metric	**Value**
Exact Match	0.000
Number of People with Max. Probability	5.512
Number of People needed to include the true values	107
Accuracy	0.015
Precision	0.004
Recall	0.026
F1 Score	0.007

Table 6.2: Baseline results of the multi-label problem with BoW format

6.2 Study of the classifiers

As previously mentioned, both SVM and NB were tested in different experiences in order to better understand if the present problem can use the same techniques of the ones studied in Chapter 3. Some of the experiences were not possible to run using the SVM classifier due to great amount of time it requires for fitting data with a lot of predictors. Hence, the SVM was only applied to experiences regarding each person's history. This section presents and compares the different results of the experiences when changing the used classifier.

Tables 6.5 and 6.6 show the results obtained by the two classifiers when running similar experiences.

Experiment	Model	F1 Score	AUC	Geo Mean
Masked data	SVM	0.064	0.500	0.000
	NB	0.306	0.503	0.472
Masked data with Pred.	SVM	0.064	0.500	0.000
	NB	0.306	0.503	0.472
BoW	SVM	0.067	0.502	0.058
	NB	0.102	0.494	0.196
TF-IDF	SVM	0.066	0.501	0.049
	NB	0.064	0.500	0.000

Table 6.3: Comparison between classifiers of each person's history results of the person with more positive instances

Experiment	Model	Accuracy	Precision	Recall
Masked data	SVM	0.030	0.001	0.056
	NB	0.297	0.008	0.415
Masked data with Pred.	SVM	0.025	0.008	0.049
	NB	0.297	0.008	0.415
BoW	SVM	0.002	0.002	0.003
	NB	0.187	0.007	0.266
TF-IDF	SVM	0.000	0.000	0.000
	NB	0.083	0.006	0.122

Table 6.4: Comparison between classifiers of people's history results of the multi-label problem

As can be seen, the Naives Bayes classifier is clearly superior for this task presenting better results in almost all metrics whereas the SVM classifier sometimes is worse than the baseline. Even when using text representations in which SVM is suppose to perform better given its robustness for text categorization, Naive Bayes obtains better results.

6.3 Study of the data construction methods

As mentioned in Chapters 4 and 5, two data construction methods were tested: one based on each person's history and other that related all tasks to all people. This section compares the results obtained by each in order to understand which provided the best results. Given that the latter method was only tested with the Naive Bayes classifier, only the result obtained with it are shown.

Tables 6.5 and 6.6 show the results obtained by the two classifiers when running similar experiences. These are identified in the list below. In the tables, PH relates to the method that considers each person's work history and CP corresponds to the Cartesian Product of all tasks and people.

- A - Using masked data
- B - Using masked data and the similarity predictor
- C - Using BoW representation
- D - Using TF-IDF representation

Experiment	Construction Method	F1 Score	AUC	Geo Mean
A	PH	0.306	0.503	0.472
	CP	0.198	0.455	0.398
B	PH	0.306	0.503	0.472
	CP	0.198	0.455	0.398
C	PH	0.102	0.494	0.196
	CP	0.107	0.424	0.286
D	PH	0.064	0.500	0.000
	CP	0.077	0.499	0.258

Table 6.5: Comparison between data construction methods using metrics regarding the individual classifiers

Experiment	Construction Method	Accuracy	Precision	Recall
A	PH	0.297	0.008	0.415
	CP	0.393	0.008	0.588
B	PH	0.297	0.008	0.415
	CP	0.393	0.008	0.588
C	PH	0.187	0.007	0.266
	CP	0.008	0.006	0.013
D	PH	0.083	0.006	0.122
	CP	0.020	0.041	0.028

Table 6.6: Comparison between data construction methods using metrics regarding the multi-label problem

It is difficulty to say with confidence that one method is superior to the other when only looking to the metrics.

6.4 Data Formats

All the studies presented in the literature review were based on text categorization. Hence there was a focus on experiences that relied on different text representations. However, evaluating and comparing all data formats can help understand the pertinence of the generated text. This section presents and compares the results obtained from the same experiences with variances on the attributes' formats or representations.

Format	F1 Score	AUC	Geo Mean
Masked	0.306	0.503	0.472
BOW	0.064	0.500	0.000
TF-IDF	0.102	0.494	0.196

Table 6.7: Comparison between data format methods using metrics regarding the individual classifiers

Format	Accuracy	Precision	Recall
Masked	0.297	0.008	0.415
BOW	0.083	0.006	0.122
TF-IDF	0.187	0.007	0.266

Table 6.8: Comparison between data formats using metrics regarding the multi-label problem

Tables 6.5 and 6.6 show the results obtained by the two classifiers when running similar experiences. Surprisingly, the experiences with the data masked are the ones that achieve the best results. They are the ones with the less information provided to the classifier and that intuitively would be less likely to be able to relate a person's skills to its tasks. This shows the text content is not the most pertinent and there should be more careful study of which attributes actually describe a task and a person.

6.5 Entity Matching Techniques

6.5.1 Similarity based Predictor

In order to better understand the value of adding the cosine similarity as a predictor to the classifiers, the best approach is to compare the classification done with masked data and only changing

that predictor.

Experiment	Model	F1 Score	AUC	Geo Mean
With Sim.	SVM	0.059	0.500	0.000
	NB	0.304	0.501	0.469
Without Sim.	SVM	0.059	0.500	0.000
	NB	0.304	0.500	0.467

Table 6.9: Comparison of the person's history results of the individual predictions of the classifiers with more than 100 positive instances with and without the similarity predictor

Experiment	Model	Accuracy	Precision	Recall
With Sim.	SVM	0.025	0.008	0.049
	NB	0.297	0.008	0.415
Without Sim.	SVM	0.030	0.001	0.056
	NB	0.297	0.008	0.415

Table 6.10: Comparison of the person history results of the multi-label problem with and without the similarity predictor

As can be seen in Table 6.9 and Table 6.10 there isn't a significant improvement when computing the similarity predictor neither for the individual classifiers nor the multi-label problem.

6.5.2 Blocking method

The blocking method used was very successful in reducing the number of negative instances and achieving better results, as shown in Tables 6.11 and 6.12.

Experiment	F1 Score	AUC	Geo Mean
With Blocking	0.263	0.500	0.475
Without Blocking	0.198	0.455	0.398

Table 6.11: Comparison of the person's history results of the individual predictions of the classifiers with more than 100 positive instances with and without the similarity predictor

Experiment	Accuracy	Precision	Recall
With Blocking	0.221	0.012	0.314
Without Blocking	0.393	0.008	0.588

Table 6.12: Comparison of the person's history results of the multi-label problem with and without the similarity predictor

6.6 Overview

As can be seen in Figures 6.1 and 6.2, the method that achieved the best results was the one that relates the tasks and people's information using the Cartesian Product and the blocking method to reduce the number of negative training instances and hence the data set imbalance. It has a good performance as a classifier because its geometric mean is one of the highest. Moreover, it is the method that can best rank the employees. It reduces the number of available people from who to choose by 50%.

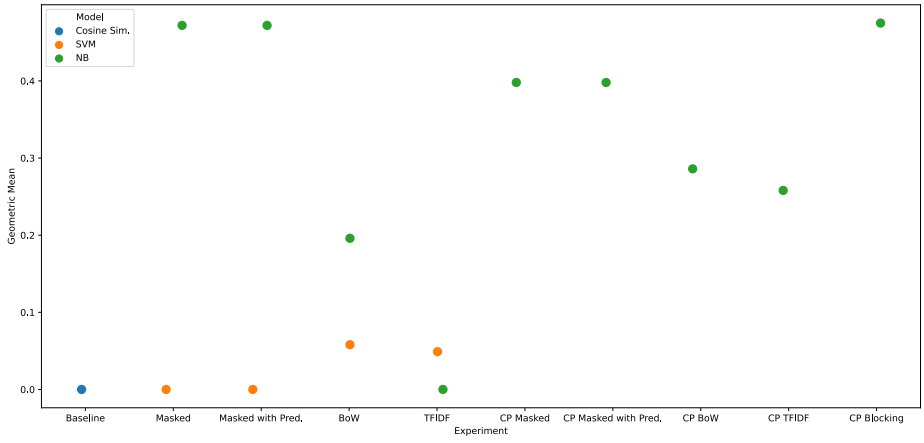

Figure 6.1: Geometric Mean of each approach

Given this problem's differences to the ones presented in the literature review, a direct comparison of the results is not relevant. However, some pertinent conclusions can be extracted. The failure of the SVM classifier for predicting good results when it was the best one in the literature review shows the impact of the imbalance in the problem. Furthermore, the BOW and TF-IDF representations weren't as successful as expected given its use in previous studies. This speaks to the quality and pertinence of the attributes. In studies of the literature review each task has a description that includes what will be done. Whereas, in this case, a task is characterized by broad fields that don't necessarily represent the specific skills they demand. Hence, it is very complex to match them to the employees profile.

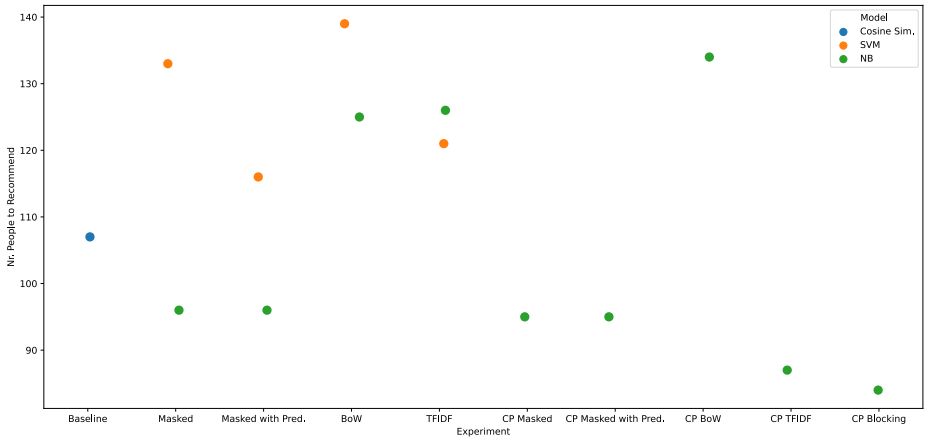

Figure 6.2: Number of people each approach has to recommend to include the correct ones

Chapter 7

Conclusions and Future Work

The present chapter concludes this research work by summarizing the research findings and answering each research question, describing its core contributions and the challenges faced. Furthermore, it also presents recommendations for future research directions.

7.1 Summary of Research Findings

This dissertation proposed an approach to the task allocation problem by studying the best methods for relating the data, formatting it, the best classifiers to be applied, and some other issues and possible methods to tackle them. Three research questions were formulated to guide the research methodology:

- **Research Question 1** - What are the best classifiers for the task assignment problem in the industry domain?
- **Research Question 2** - What are the best methods for data construction when relating two entities?
- **Research Question 3** - What are the main issues of this problem and which methods can be applied to improve it?

The following points justify that each research question was answered and, hence, this dissertation goal was accomplished.

7.1.1 What are the best classifiers for the task assignment problem in the industry domain?

As mentioned in Chapter 4, two classifiers were applied: SVM and Naive Bayes. The former was used because it presented the best results in the problems of the literature review. The latter

was chosen due to its simplicity, which makes it more efficient for classification problems with a significant amount of instances and predictors, and decent results for text categorization [17].

In the experiments displayed in Chapter 5, Naive Bayes showed the best results overall, having SVM not even proved to be better than the baseline in most cases. The imbalance specific metrics such as the geometric mean showed that SVM struggles when predicting the minority class. In other words, its performance is unsatisfactory when allocating people to a task, leading to low accuracy when evaluating the global multi-label problem.

SVMs issues when used in imbalanced data sets are explained by Yanmin et al. [34]. It says that SVMs learn to classify all instances as the majority class to make the margin as large as possible and minimize the error. Hence, it has many difficulties predicting the minority class.

Furthermore, Naive Bayes proved to be more valuable than SVM due to its efficiency when running the experiences. The former lasted a lot less time to train and test the models, which enabled experiences with more instances and predictors to be run. This can be crucial when a new model has to be trained from scratch (in case of adding or deleting a person, for example), especially in an industry domain.

Therefore, it can be concluded with confidence that Naive Bayes is superior to SVM for the problem in question.

7.1.2 What are the best methods for data construction when relating two entities?

A crucial aspect of the task allocation problem that sets it apart from any other classification problem is the need to relate two entities described by attributes representing different information.

Firstly, it was essential to define the problem as a classification one. Like many works in the literature review, it is a multi-label problem where the people are the classes to be assigned, and the tasks are the instances provided to the classifier. The former was chosen from the two possible transformations available (problem or method) due to its great use in literature review and its simplicity, making it easier for the two entities' attributes to be used as predictors.

Secondly, given that the problem transformation was chosen, two methods for data construction were used. One solely was based on each person's history. The predictors corresponded to only the tasks' attributes, and the classifier had to decide whether or not that person was appropriate for that task. The other method was the Cartesian product of all tasks and users.

The first one relies on what a person has done in the company. A model is computed per person, and the framework's efficacy depends on the performance of each model. With this method, it is harder to add people to the framework. If they do not have any previous history, the model will not have any data on which to base its decision. The lack of data corresponds to a common issue in recommender systems - the cold start problem [22]. Applying a blocking method to reduce the data set imbalance intensifies this problem, given that there will be fewer instances with which to train a person's history. This data construction method is not the most appropriate one for this problem because either the data set is an imbalance or there is insufficient data to train a model.

The second method depends on the relationship between a task's and a person's attributes. All data is provided at the same time, and the model is evaluated as one. This method obtained the

best results when combined with blocking techniques. It seems to be the most appropriate data construction practice for this problem. It does not have to deal with the cold start problem, has as good results as the others in almost all experiences, and achieves the best performance with the blocking method.

7.1.3 What are the main issues of this problem and which methods can be applied to improve it?

Many similarities to the entity matching problems could be pointed out when defining this problem and studying a possible approach to match a task and a user. They both need to relate attributes of two different entities, resulting in a significant imbalance of the data set and a high number of predictors.

The approach used to relate the two entities in one attribute did not hinder or help the classifier, given that its results were very similar to other experiences. This could be due to the lack of value when comparing two texts that describe very different attributes or the hardship of representing a unique value of two texts' complexity and full meaning. The similarity of more specific attributes such as the task's and the person's department instead of comparing a whole text could bring more precise information to the classifier and improve it's performance.

On the other hand, the blocking method successfully reduced the number of instances to be compared in the Cartesian Product. It improved the results, which could mean it is a step in the right direction towards handling the data imbalance inherent to this kind of problem.

Finally, the text generated from the categorical or numerical attributes performed worse than the masked attributes. This could be a reflection of the text's content pertinence and quality. It is rather hard to correspond a project's and a task's title, which are usually broad and vague fields, to the specific skills they demand. Hence there is a need for a task description in order for that information to be included.

7.1.4 Hypothesis

"Machine learning methods can be applied for task assignment and lead to meaningful results that help triage appropriate people for a given task"

The previous research questions proved that some machine learning methods can be useful when used to triage people for a task. When compared those results with our baseline, a method that does not use machine learning, almost all approaches improved it.

7.2 Future Work

The proposed approach could help the decision making process of task allocation by slightly reducing the number of people from whom to choose. However, this triage could be further improved, and several paths could be taken.

Firstly, there could be a more careful filter and storage of the data to ensure which users are employed by the company and available for each task. Furthermore, some people are perfectly capable of performing tasks, but there is no way of knowing it. The only stored information is whether or not someone was allocated to a task, not if someone was appropriate to do it. For example, if two people share the same profile and a task only needs one person with that curriculum, solely one of them will be allocated to that task. There is no information in the database that other people could also be a possibility. Hence, when training the classifier, if there are two people with same profile, they will have the same attributes. However, they will be classified as opposites because only one was allocated to do it. This greatly hinders the performance of any classifier. So, as in many other classification problems, a more extensively annotated data set to train the models could mean a good improvement in the approach.

Although Naive Bayes proved to be the superior classifier for this approach, other classifiers could outperform it. Associating Bagging, Boosting, and Random Forests with sampling or cost-sensitive methods prove to be highly competitive and robust to complex data [19]. Under and oversampling methods are also an excellent way to handle imbalance data, also in multi-label problems [8].

Furthermore, a non-binary approach to the problem transformation or even using method transformation that multi-label demands could provide good results [27].

7.3 Conclusion

A lot of the decision making processes in the industry domain has been looking for support in machines in order to reduce the risk for mistakes due to subjectivity or lack of knowledge. An important process is the task allocation which requires the manager to be aware of a lot of information. This process has not been the subject a lot of research and could mean the improvement of decisions that influence the whole company. The present dissertation studied different classifiers and techniques to apply to this kind of problem and proposed a method that can serve as a foundation for future work.

Appendix A

Experiences

A.1 User history based Results

The present section presents the several combinations for the experiences done using the users' task history.

Masked data

Table A.1 presents the results obtained when using a classifier per user with the task's attributes masked. Table A.2 presents the results obtained from the same experiences but evaluating through the multi-label lenses.

	Model	F1 Score	AUC	Geo Mean
User A	SVM	0.064	0.500	0.000
	NB	0.306	0.503	0.472
Mean	SVM	0.013	0.501	0.003
	NB	0.257	0.457	0.232
Top Mean	SVM	0.059	0.500	0.000
	NB	0.304	0.500	0.467

Table A.1: User history results of the individual predictions with masked format

Metric	SVM	Naive Bayes
Exact Match	0.000	0.000
Number of Users with Max. Probability	6.682	92.593
Number of Users needed to include the true values	133	96
Accuracy	0.030	0.297
Precision	0.001	0.008
Recall	0.056	0.415
F1 Score	0.002	0.016

Table A.2: User history results of the multi-label problem with masked format

Masked data and Similarity Predictor

Table A.3 presents the results obtained when using a classifier per user with the task's attributes masked and the similarity predictor. Table A.4 presents the results obtained from the same experiences but evaluating through the multi-label lenses.

	Model	F1 Score	AUC	Geo Mean
User A	SVM	0.064	0.500	0.000
	NB	0.306	0.503	0.472
Mean	SVM	0.012	0.500	0.001
	NB	0.256	0.457	0.232
Top Mean	SVM	0.059	0.500	0.000
	NB	0.304	0.501	0.469

Table A.3: User history results of the individual predictions with masked format and the similarity predictor

Metric	**SVM**	**Naive Bayes**
Exact Match	0.000	0.000
Number of Users with Max. Probability	7.037	92.614
Number of Users needed to include the true values	116	96
Accuracy	0.025	0.297
Precision	0.008	0.008
Recall	0.049	0.415
F1 Score	0.002	0.016

Table A.4: User history results of the multi-label problem with masked format and the similarity predictor

BoW Representation

Table A.5 presents the results obtained when using a classifier per user with the task's attributes represented using BoW. Table A.6 presents the results obtained from the same experiences but evaluating through the multi-label lenses.

	Model	**F1 Score**	**AUC**	**Geo Mean**
User A	SVM	0.067	0.502	0.058
	NB	0.102	0.494	0.196
Mean	SVM	0.010	0.452	0.010
	NB	0.022	0.469	0.062
Top Mean	SVM	0.060	0.497	0.022
	NB	0.081	0.491	0.140

Table A.5: User history results of the individual predictions with BoW representation

Metric	SVM	Naive Bayes
Exact Match	0.000	0.000
Number of Users with Max. Probability	2.765	105.583
Number of Users needed to include the true values	139	125
Accuracy	0.002	0.187
Precision	0.002	0.007
Recall	0.003	0.266
F1 Score	0.002	0.008

Table A.6: User history results of the multi-label problem with BoW representation

TF-IDF Representation

Table A.7 presents the results obtained when using a classifier per user with the task's attributes represented using TF-IDF. Table A.8 presents the results obtained from the same experiences but evaluating through the multi-label lenses.

	Model	F1 Score	AUC	Geo Mean
User A	SVM	0.066	0.501	0.049
	NB	0.064	0.500	0.000
Mean	SVM	0.010	0.447	0.011
	NB	0.010	0.493	0.000
Top Mean	SVM	0.061	0.497	0.035
	NB	0.059	0.500	0.000

Table A.7: User history results of the individual predictions with TF-IDF representation

Metric	SVM	Naive Bayes
Exact Match	0.000	0.000
Number of Users with Max. Probability	3	48.733
Number of Users needed to include the true values	121	126
Accuracy	0.002	0.083
Precision	0.002	0.006
Recall	0.002	0.122
F1 Score	0.002	0.002

Table A.8: User history results of the multi-label problem with TF-IDF representation

A.2 Cartesian Product based Results

	F1 Score	AUC	Geo Mean	Accuracy
Value	0.198	0.455	0.398	0.239

Table A.9: Cartesian Product results of the individual predictions with masked representation

Metric	Value
Exact Match	0.000
Number of Users with Max. Probability	120.149
Number of Users needed to include the true values	95.074
Accuracy	0.393
Precision	0.008
Recall	0.588
F1 Score	0.017

Table A.10: Cartesian Product results of the multi-label problem with masked representation

	F1 Score	**AUC**	**Geo Mean**	**Accuracy**
Value	0.198	0.455	0.398	0.239

Table A.11: Cartesian Product results of the individual predictions with masked representation and the similarity score

Metric	**Value**
Exact Match	0.000
Number of Users with Max. Probability	120.167
Number of Users needed to include the true values	95.090
Accuracy	0.393
Precision	0.008
Recall	0.588
F1 Score	0.017

Table A.12: Cartesian Product results of the multi-label problem with masked representation and the similarity score

	F1 Score	**AUC**	**Geo Mean**	**Accuracy**
Value	0.263	0.500	0.475	0.345

Table A.13: Cartesian Product results of the individual predictions with masked representation using the blocking method

Metric	**Value**
Exact Match	0.000
Number of Users with Max. Probability	47
Number of Users needed to include the true values	84
Accuracy	0.221
Precision	0.012
Recall	0.314
F1 Score	0.022

Table A.14: Cartesian Product results of the multi-label problem with masked representation using the blocking method

	F1 Score	**AUC**	**Geo Mean**	**Accuracy**
Value	0.107	0.424	0.286	0.117

Table A.15: Cartesian Product results of the individual predictions with BoW representation

Metric	**Value**
Exact Match	0.000
Number of Users with Max. Probability	3.000
Number of Users needed to include the true values	134.779
Accuracy	0.008
Precision	0.006
Recall	0.013
F1 Score	0.007

Table A.16: Cartesian Product results of the multi-label problem with BoW representation

	F1 Score	**AUC**	**Geo Mean**	**Accuracy**
Value	0.077	0.499	0.258	0.080

Table A.17: Cartesian Product results of the individual predictions with TF-IDF representation

Metric	**Value**
Exact Match	0.012
Number of Users with Max. Probability	2.625
Number of Users needed to include the true values	87.204
Accuracy	0.020
Precision	0.041
Recall	0.028
F1 Score	0.031

Table A.18: Cartesian Product results of the multi-label problem with TF-IDF representation

References

[1] Charu C. Aggarwal. *Machine Learning for Text*. Springer Publishing Company, Incorporated, 1st edition, 2018.

[2] John Anvik, Lyndon Hiew, and Gail C. Murphy. *Who Should Fix This Bug?*, page 361–370. Association for Computing Machinery, New York, NY, USA, 2006.

[3] Michael Arias, R. Saavedra, Maíra R. Marques, J. Munoz-Gama, and M. Sepúlveda. Human resource allocation in business process management and process mining. *Management Decision*, 56:376–405, 2018.

[4] Ana Barcus and Gilberto Montibeller. Supporting the allocation of software development work in distributed teams with multi-criteria decision analysis. *Omega*, 36:464–475, 06 2008.

[5] Sana Bouajaja and Najoua Dridi. A survey on human resource allocation problem and its applications. *Operational Research*, 17(2):339–369, 2017.

[6] D.C. Brabham. *Crowdsourcing*. The MIT Press Essential Knowledge series. MIT Press, 2013.

[7] Dolly Carrillo, Vivian F. López, and María N. Moreno. Multi-label classification for recommender systems. In Javier Bajo Pérez, Juan M. Corchado Rodríguez, Johannes Fähndrich, Philippe Mathieu, Andrew Campbell, Mari Carmen Suarez-Figueroa, Alfonso Ortega, Emmanuel Adam, Elena Navarro, Ramon Hermoso, and María N. Moreno, editors, *Trends in Practical Applications of Agents and Multiagent Systems*, pages 181–188, Cham, 2013. Springer International Publishing.

[8] Francisco Charte, Antonio J Rivera, María J del Jesus, and Francisco Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16, 2015.

[9] Lydia B. Chilton, John J. Horton, Robert C. Miller, and Shiri Azenkot. Task search in a human computation market. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, page 1–9, New York, NY, USA, 2010. Association for Computing Machinery.

[10] K.R. Chowdhary and W. Bibel. *Fundamentals of Artificial Intelligence*. Springer India, 2020.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[12] J. Eisenstein. *Introduction to Natural Language Processing*. Adaptive Computation and Machine Learning series. MIT Press, 2019.

[13] Vitor Miguel Saraiva Esteves. Techniques to deal with imbalanced data in multi-class problems: A review of existing methods. 2020.

[14] Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Knowledge Discovery in Databases: PKDD 2006*, pages 503–510, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[15] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. volume vol. 3056, 08 2004.

[16] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, SIGMOD '95, page 127–138, New York, NY, USA, 1995. Association for Computing Machinery.

[17] Ashraf M. Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In Geoffrey I. Webb and Xinghuo Yu, editors, *AI 2004: Advances in Artificial Intelligence*, pages 488–499, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[18] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4), 2019.

[19] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[20] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data Knowledge Engineering*, 69:197–210, 02 2010.

[21] Sun-Ro Lee, Min-Jae Heo, Chan-Gun Lee, Milhan Kim, and Gaeul Jeong. Applying deep learning based automatic bug triager to industrial projects. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 926–931, New York, NY, USA, 2017. Association for Computing Machinery.

[22] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[23] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[24] Ke Mao, Ye Yang, Qing Wang, Yue Jia, and Mark Harman. Developer recommendation for crowdsourced software development tasks. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 347–356, 2015.

[25] Prem Melville and Vikas Sindhwani. Recommender systems. *Encyclopedia of machine learning*, 1:829–838, 2010.

[26] Pedro Miranda, J. Pascoal Faria, Filipe F. Correia, Ahmed Fares, Ricardo Graça, and João Mendes Moreira. An analysis of monte carlo simulations for forecasting software projects. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, SAC '21, page 1550–1558, New York, NY, USA, 2021. Association for Computing Machinery.

[27] Jose M Moyano, Eva L Gibaja, Krzysztof J Cios, and Sebastián Ventura. Review of ensembles of multi-label classifiers: models, experimental study and prospects. *Information Fusion*, 44:33–45, 2018.

[28] Hoda Naguib, Nitesh Narayan, Bernd Brügge, and Dina Helal. Bug report assignee recommendation using activity profiles. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, page 22–30. IEEE Press, 2013.

[29] Nicola Paltrinieri, Louise Comfort, and Genserik Reniers. Learning about risk: Machine learning for risk assessment. *Safety Science*, 118:475–486, 2019.

[30] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. A survey of blocking and filtering techniques for entity resolution. *CoRR*, abs/1905.06167, 2019.

[31] Robert Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning - ML*, 39:135–168, 05 2000.

[32] Saad Shafiq, Atif Mashkoor, Christoph Mayr-Dorn, and Alexander Egyed. Taskallocator: A recommendation approach for role-based tasks allocation in agile software development. *CoRR*, abs/2103.02330, 2021.

[33] Mohammad S. Sorower. A literature survey on algorithms for multi-label learning. 2010.

[34] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, 23(04):687–719, 2009.

[35] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 09 2009.

[36] Grigorios Tsoumakas, Ioannis Katakis, and I. Vlahavas. *Mining Multi-label Data*, pages 667–685. 07 2010.

[37] R. Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 01 2000.

[38] Wenjin Wu, Wen Zhang, Ye Yang, and Qing Wang. Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking. In *2011 18th Asia-Pacific Software Engineering Conference*, pages 389–396, 2011.

[39] Xin Xia, David Lo, Xinyu Wang, and Bo Zhou. Accurate developer recommendation for bug resolution. pages 72–81, 10 2013.

[40] Xin Xia, David Lo, Xinyu Wang, and Bo Zhou. Dual analysis for recommending developers to resolve bugs. *Journal of Software: Evolution and Process*, 27, 03 2015.

[41] Ye Yang, Muhammad Rezaul Karim, Razieh Saremi, and Guenther Ruhe. Who should take this task? dynamic decision support for crowd workers. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '16, New York, NY, USA, 2016. Association for Computing Machinery.

[42] Yue Yu, Huaimin Wang, Gang Yin, and Charles X. Ling. Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration. In *2014 21st Asia-Pacific Software Engineering Conference*, volume 1, pages 335–342, 2014.

[43] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology*, 74, 01 2016.

[44] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.