UNIVERSITY OF NAPLES

FEDERICO II

Master of Science course of Computer Science

Department of Electrical Engineering and Information Technology

Information Retrieval System course paper

Ontology-based Text Classification

Professor:

Antonio Maria Rinaldi

Student:

Benfenati Domenico M63001165

ACADEMIC YEAR 2021/2022

Second Semester

Contents

Chapter 1

Introduction

Document classification or **document categorization** is a problem in library science, information science and computer science. The task is to assign a document to one or more classes or categories. This may be done "**manually**" or **algorithmically**[1].

Text classification techniques are used in many applications, including e-mail filtering, mail routing, spam filtering, news monitoring, sorting through digitized paper archives, automated indexing of scientific articles, classification of news stories and searching for interesting information on the web, biomedical applications, etc. However, it is often the case that a suitable set of well classified trained corpus is not available. Even if one is available, the set may be too small, or a significant portion of the corpus in the training set may not have been classified properly. This creates a serious limitation for the usefulness of the traditional text classification methods.

Text classification has been one of the most sought after and researched areas in machine learning. Even though it's applications are numerous, it still is a lengthy process to go through. One of the best strategies to identify the category of a text is to take advantage of some knowledge representation, such as an **ontology**. In computer science, an **ontology** is a formal, shared and explicit representation of a conceptualisation of a domain of interest[2]. More specifically, it is a first-order axiomatic theory expressible in descriptive logic.

The first chapter of this work describes the classification methodologies in place and the approach used for classification using ontologies.

The second chapter describes the implementation details of the solution to the classification problem and provides the results.

The last chapter describes the comparisons made between classical techniques and the one proposed in this work.

Chapter 2

Ontology-based Classification

As the amount of textual information available electronically continues to grow, organizations face the challenge of organizing, analyzing, and extracting knowledge from large amounts of unstructured information for decision making.

Traditional classification approaches use statistical or machine learning methods to accomplish this task. Such methods include Naıve Bayes, Support Vector Machines, Latent Semantic Analysis, and many others. All of these methods require a training set of pre-classified documents that are used to train a classifier, which can then correctly assign categories to other previously unseen documents.

So far, however, little research has been done on integrating semantic background knowledge into text classification. They have used **WordNet** to improve text clustering tasks. **WordNet**[3] is a network of related words organized into sets of synonyms, each set representing one basic lexical concept. Classifying texts using semantic concepts is a step from classification based on simple words and phrases to classification based on *semantics*. Latent Semantic Analysis provides an attractive way to move from the space of words to the space of concepts of related phrases.

Another approach is based on **ontologies**. Ontologies provide more structurally and semantically organized knowledge. The knowledge expressed in a complex ontology can be used to identify concepts in a text. Moreover, if the concepts in the ontology are organized into a hierarchy of higher-level categories, it should be possible to identify the categories that best categorize the content of the text. Because ontologies provide named entities or terms and the relationships between them, the intermediate categorization step requires mapping terms to ontology entities.

Another approach is to reinforce the co-occurrence of certain words or pairs of entities linked in the ontology in the term vector. Here, classification can be based on named entities that have been recognized and parsed. Traditional classification methods can also be improved with ontology-based information about the neighborhood of entities. However, these ontology-based approaches are still insufficient in terms of classification accuracy. As mentioned above, the novelty of the approach in this work compared to traditional classification methods is that the classification method does

not require a training set, unlike traditional statistical and probabilistic methods. In addition, the proposal method uses a thesaurus to find implicit connections between the classified text and ontology terms. This feature extends the applicability of the classifier to complex real-world texts and makes it robust to ontology imperfections.

Chapter 3

Knowledge Background

This chapter aims to make known current text classification approaches and the procedures upstream of these algorithms, such as data pre-processing procedures.

3.1 Pre-Processing Techniques

According to the official documentation, the Natural Language Toolkit (NLTK)[4] is a platform used to create Python programs that process human language data for use in statistical natural language processing (NLP). NLTK is a useful Python tool that provides algorithms for processing a wide range of languages. This tool is powerful because it is free and open source. In addition, the official NLTK documentation is very well written, so there is no need to look for special tutorials when using NLTK. The most common algorithms used in NLTK include tokenization, lemmatization and part-of-speech tagging. These algorithms are mainly used for preprocessing text data. The preprocessing is divided into five parts:

- **Tokenization**: Tokens are the basic building blocks of any linguistic structure such as sentences and paragraphs. The tokenization process[5] is the decomposition of these structures into tokens. There are two types of tokenization: a paragraph becomes the unit that can be broken down into different tokens, which are the sentences; more detailed, however, is the breakdown at sentence level into words.

- **Stemming**: Stemming a word is the root or phrase from which various forms of a word are derived. Stemming is the process of identifying all words formed from the same root and reducing or normalizing them to their root form. For example, the words ”connection”, ”connected”, and ”connecting” are reduced to the common word *connect*.

- **Lemmatization**: In some cases, there may be words with different roots but the same final meaning. In such cases, it is necessary to consult a dictionary to further narrow the stem down to a common meaning, i.e., a base word. This base word is called lemma and is known

as lemmatization. For example, the word "better" has "good" as a lemma. Since the two words are not exactly the same, they are excluded by lemming, and you have to consult a dictionary to check their meaning in the lemma.

- **Part-of-speech tagging**: denotes part-of-speech that, as the name implies, identifies different parts of a linguistic structure, such as sentences. Parts of speech include adjectives, nouns, and verbs. They are identified by examining sentence structure and observing the arrangement of words and the connections between them. This is done by examining sentence structure and observing the arrangement of words and the connections between them.

3.2 Text classification Techniques

The idea behind text classification is to use machine learning to classify text into categories. This has been used in many related fields such as sentiment analysis, emotional analysis, etc. Several classifiers have been developed for each classification category. Text classifiers are created with the intention of being implemented on a variety of text data sets. Text classifiers work with both structured and unstructured datasets. Both types of datasets find numerous applications in a wide variety of fields. The classification process in machine learning can be described very simply. First, the training dataset is evaluated and analyzed for boundary conditions. Next, classes are predicted for new data using information learned and learned during the training phase. This is the entire classification process.

Classification can be supervised or unsupervised. Supervised classification works on the principles of learning and testing and uses labeled data, i.e., predefined classes, to make predictions. In the training phase, the model is trained on some predefined classes by feeding it labeled or labeled data[6]. In the testing phase, the model's prediction or classification performance is measured by feeding it unobserved data. In other words, only the classes learned in the training phase can be predicted in the testing phase.

Common examples of supervised classification are spam filtering and intentional detection.

In unsupervised classification, classification is performed by the model without any external information. In this case, the model algorithm attempts to group or cluster data points based on similar characteristics, patterns, and other common attributes that can be used to link two data points[7]. A common example where classification without observation can be really useful is in search engines. Search engines create clusters of data based on data from previous searches.

This type of classification is highly customizable and dynamic because there is no need to train or label text data sets. Thus, unsupervised classification is compatible with languages. The classifiers used to classify texts are ML classifiers based on classifiers (e.g., Naive Bayes classifiers, decision tree classifiers) or neural network based classifiers (e.g., artificial neural networks, convolutional neural networks).

The basic text classification algorithm consists of three modules, schematized in Figure 3.1:

- The first step is to read and pre-process, if necessary, the input dataset one chooses to use. This is a fairly important step in the whole process: depending on the level of pre-processing applied to the dataset, the experimental results obtained may vary: for example, if one decides to carry out a stemming procedure against a lemmatisation, one must take into account that stemming may produce some non-existent words since the root of the extracted word does not necessarily correspond to the morphological one.

- The second step is to apply machine learning algorithms to the dataset thus processed; as the dataset initially contains text keywords, it needs to be converted into numbers. Conversion into numbers can be done by various techniques, the most commonly used being the one that binds the textual string to a numeric matrix according to various methods. Some libraries already provide such conversion with pre-implemented algorithms, as in the case of the CountVectorized function of the Python library Sklearn[8]. After this the training data is ready for feeding to the classifier for training. After training, we can use the model for predictions on testing data.

- The final step is the analysis of the classification, using different metrics such as accuracy, precision, recall or F-measure[9]. All these metrics are aimed at evaluating what is referred to as the performance of the classification algorithm, because they are based on the correct or incorrect classification of instances unknown to the algorithm, from the so-called test set.

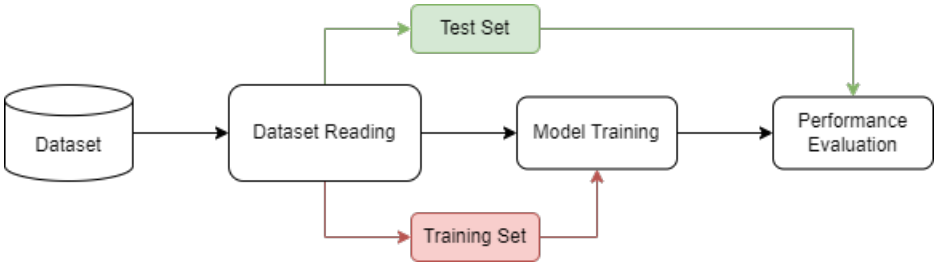

Figure 3.1: Basic Text Classification schema using ML

Chapter 4

Proposed Approach

This chapter describes the proposed procedure for classification using ontologies, in particular how machine learning approaches can be aided by integrating an ontology into the process. The aim is to verify this improvement by comparing the two methodologies in terms of performance metrics.

4.1 Datasets Used

With regard to the knowledge base used, we chose to use a knowledge database of disease-symptom associations generated by an automated method based on information in textual discharge summaries of patients at New York Presbyterian Hospital, called **Disease-Symptom Knowledge Database**[10].

The first column shows the disease, the second the number of discharge summaries containing a positive and current mention of the disease, and the associated symptom. Associations for the 150 most frequent diseases based on these notes were computed and the symptoms are shown ranked based on the strength of association.

The method used the MedLEE natural language processing system to obtain UMLS codes for diseases and symptoms from the notes; then statistical methods based on frequencies and co-occurrences were used to obtain the associations.

For the ontology-based approach, we use **Human Disease Ontology**[11]. The Disease Ontology has been developed as a standardized ontology for human disease with the purpose of providing the biomedical community with consistent, reusable and sustainable descriptions of human disease terms, phenotype characteristics and related medical vocabulary disease concepts through collaborative efforts of biomedical researchers, coordinated by the University of Maryland School of Medicine, Institute for Genome Sciences.

The Disease Ontology semantically integrates disease and medical vocabularies through extensive cross mapping of DO terms to MeSH, ICD, NCI's thesaurus, SNOMED and OMIM.

4.2 Conventional Classification

The conventional text classification framework is based on three main modules, as mentioned in the previous chapters: dataset creation and eventual preprocessing, training and testing of the machine learning model, and analysis of the results, as shown in Figure 3.1.
In order to use the dataset mentioned before for the study, changes had to be made. In addition, new information was added to the dataset to ensure accuracy of comparison. The final dataset thus created is the dataset used in the proposed study. The dataset modification operation was necessary in order to make the ontology and the output of the classification operation compatible, as the 'disease name' feature within the ontology was made compliant with the otuput feature of the applied classification model. Then the dataset processing procedure was carried out using the NTLK library. Synthetic datasets are also created using programming techniques to create new data. In this study, multiple records were created by randomly selecting traits of the same class. For example, suppose a disease has 10 symptoms. A subset of those 10 symptoms is randomly selected to create a new record in the dataset containing fewer symptoms and disease names. This process associates symptom values with disease names and creates a strong positive relationship between feature values (symptoms) and classes (disease names).
The dataset elaboration process was descripted in Figure 4.1.

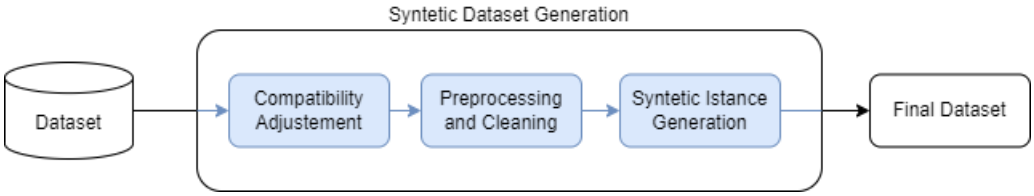

Figure 4.1: Dataset Elaboration Pipeline

After processing the dataset, machine learning must be brought into play to train the models and make predictions on that dataset.
As explained in the previous chapter, since the data at stake are words, it is necessary to transform these words into numbers by implementing a type of encoding. We chose to make use of the CountVectorized module to map each word within a matrix of features that the classifier will use for its training and to perform the classification itself.
The classifiers used include KNN,SVM,logistic regression,decision trees, Random forests,etc. After training,the model can be used to predict test data. The ratio chosen of training and test data is 80 and 20,respectively.
The model training and test process was descripted in Figure 4.2.
Having obtained the predictions of the model, all that remains is to perform the performance analysis based on the predictions. To analyze the results, the disease predictions for the test data are compared to the actual disease classes. After the comparison, classification metrics such as accu-

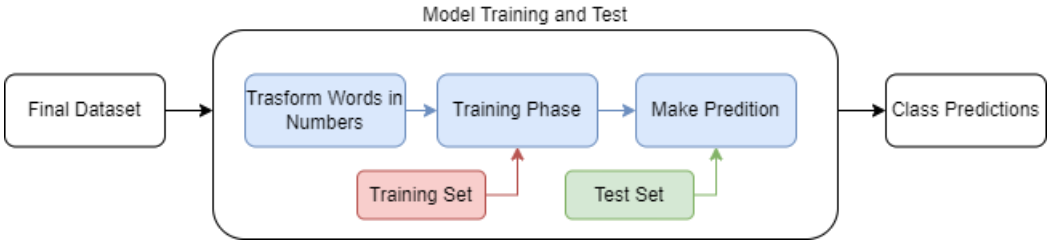

Figure 4.2: Model Train and Test Pipeline

racy, precision, repeatability and F1 score are calculated. After this calculation, the performance of several classifiers based on the metrics can be compared.

The results of the comparison can be compared. accuracy and recall values for each class to see which class performs better.

The analysis evaluation pipeline is shown in Figure 4.3.

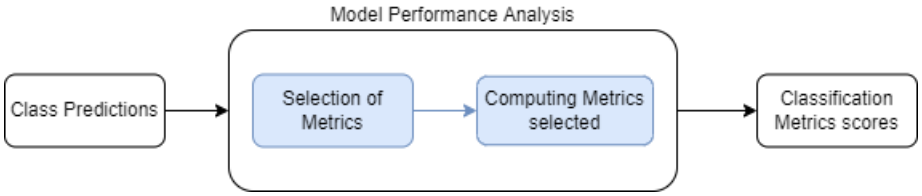

Figure 4.3: Model Performance Analysis Pipeline

4.3 Ontology-based Classification

Unlike the classification approach described above, the use of ontology introduces an additional step in the classification process, prior to the training of the machine learning model, which is called **Ontology Matching**.

This phase consists of different steps, schematised in Figure 4.4.

The keywords generated from the disease descriptions are mapped to the keywords in the ontology nodes. All mapped nodes are potential classes that can be used to create a subset of data for efficient model training. Classes can be further constrained using priority-based mapping. In this study, for ontology matching, two numbers are assigned to each keyword to determine its priority. The first number represents the frequency of the keyword, and the second indicates whether or not the keyword is rematable.

Thus, each keyword has a syntax `(name, first priority number, second priority number)`.

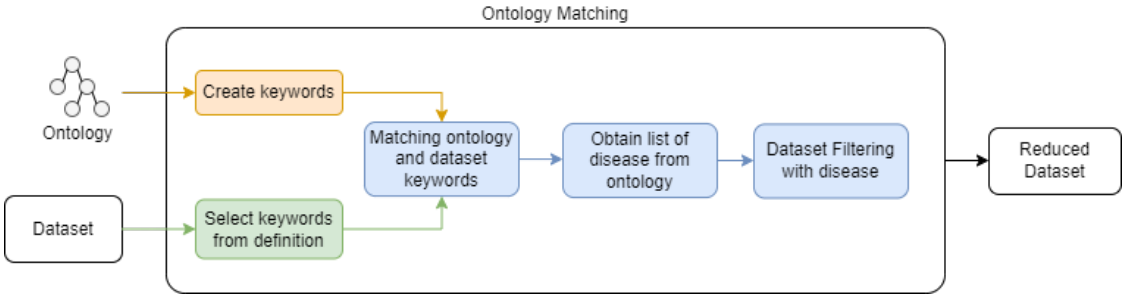

Figure 4.4: Ontology Matching Process

The new schema of classification is exposed in Figure 4.5.

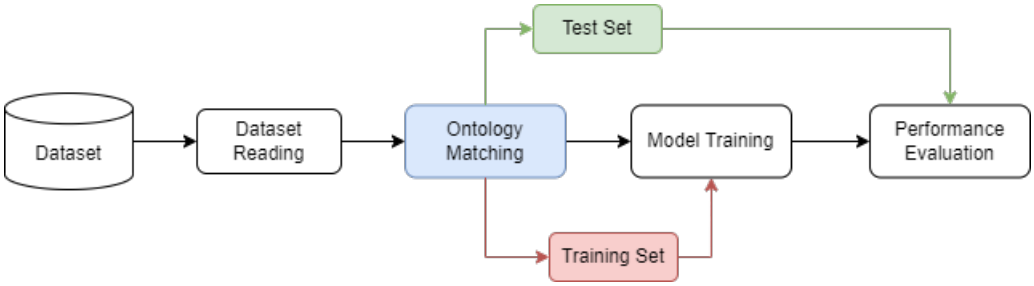

Figure 4.5: Ontology-based Classification Process

Chapter 5

Implemented Solution

The implementation details of the solution described above are proposed below, both in terms of pseudocode and the actual implementation in Python language.

5.1 Classification without Ontology

First, we need to define what type of classifiers we have to use. After defined a list of this models, we can perform training of each of them with the data in input from the pre-processed dataset, as told come paraghaph before.
Once the classifiers have been trained, we can perform the evaluation phase, applying all models on the test istances, in order to make the predictions.

Algorithm 1 ClassificationWithoutOntology

Require: *(x,y)train, (x,y)test*: train and test datasets, split in data(`x`) and class(`y`)

Ensure: *predictions*: list of predictions made from classifiers

```
1: classifiers ← list of predefined classifiers
2: _train ← CountVectorized(xtrain)
3: _test ← CountVectorized(xtest)
4: predictions ← []
5: for all model ∈ classifiers do
6:     Inizialize(model)[1]
7: end for
8: for all model ∈ classifiers do
9:     Training(model, ytrain)
10:    predictions ← predictions ∪ MakePrediction(model,_test)
11: end for
12: return predictions
```

The ordinary classification algorithm can be applied to the workflow described in the previous chapter, the pseudocode of which is given below. First, the dataset must be loaded, processing all its instances as described above. We then proceed to split this dataset into training and test subsets. Finally, after applying the classification algorithm to the subsets just created, we move on to the evaluation phase, which consists of calculating classification metrics such as accuracy, precision, recall, and F-measure for each of the classifiers used.

Algorithm 2 process-without-ontology

Ensure: *metrics*: list of metrics calculated for all classifiers

1: dataset ← DatasetLoading[2]()
2: classifiers ← define a list of used classifiers
3: metrics ← []
4: train,test ← DatasetSplit[3](dataset)
5: predictions ← ClassificationWithoutOntology(train,test)
6: **for all** c ∈ classifiers **do**
7: metrics ← metrics ∪ Metrics[4](c)
8: **end for**
9: **return** metrics

[1]The Initialize, Training and MakePrediction functions refers to that of python's *Sklearn* library of all classifiers that we decided to use.

[2]The function DatasetLoading takes the classical dataset as input, pre-processes it, and proceeds to augment the instances, creating a synthetic dataset as described in the previous chapter.

[3]The function DatasetSplit is offered by the Sklearn library and divides the dataset into train and test, according to a percentage scheme decided a priori, and outputs the test and train vectors separated by their respective classes.

[4]The function Metrics refers to the calculation of user-defined evaluation metrics. You can take advantage of the metrics calculation functions offered by the Sklearn library.

5.2 Classification with Ontology

Algorithm 3 ClassificationWithOntology

Require: *(x,y)train, (x,y)test*: train and test datasets, split in data(x) and class(y)

Ensure: *predictions*: list of predictions made from classifiers

```
Synth_dataset(x,y) ← Create a Synthetic dataset from the input one
all_classes ← Synth_dataset_y
ontology ← OntologyLoading()
classifiers ← list of predefined classifiers
for all istance ∈ xtest do
    keywords ← Select keyword from Knowledge Base
    poss_classes ← OntologyMatching(ontology,keywords)
    final_class ← []
    for all onto_class, prev_class ∈ poss_classes, all_classes do
        if prev_class IsSubsetOf onto_class then
            final_class ← final_class ∪ prev_class
        end if
    end for
    if final_class = [] then
        predictions ← ClassificationProcess((x,y)train, istance)
    else
        instanceToUse ← []
        for all train_ist ∈ ytrain do
            if train_ist ∈ final_class then
                istanceToUse ← istanceToUse ∪ train_ist
            end if
        end for
        predictions ← ClassificationProcess(istanceToUse, istance)
    end if
end for
return predictions
```

The ontology classification process takes train and test datasets as input and implements the methodology described above for classification. Classification is done using the traditional train dataset if the instance is not contained within the common classes with the ontology, while the classification process uses the reduced dataset if the instance is matched within the ontology. The matching procedure is described below.

Algorithm 4 OntologyMatching

Require: *tree*: the ontology processed

keys: keywords to search into the ontology

Ensure: *poss_classes*: list of possible classes from the ontology

```
for all key ∈ keys do
  if key.Priority ≠ 1 then
    prio1_classes ← prio1_classes ∪ key
  else
    if key.Lemmatizable = 1 then
      prio2_classes ← prio2_classes ∪ key
    else
      prio3_classes ← prio3_classes ∪ key
    end if
  end if
end for
for all elem ∈ tree do
  prio1_count ← Count how much prio1_classes are in elem.Keywords
  prio2_count ← Count how much prio2_classes are in elem.Keywords
  prio3_count ← Count how much prio3_classes are in elem.Keywords
  if prio1_count + prio2_count + prio3_count ≥ 1 then
    poss_classes ← poss_classes ∪ elem.Entity
  end if
end for
return poss_classes
```

It was chosen to assign a higher priority to those terms that had an immediate correspondence with the entities in the ontology, while a variable average priority was assigned depending on whether the term is lemmatisable or not: if the word is already a lemma, it is assigned priority 2, and if not, it is assigned priority 3. If, among all the terms with the different priorities, a match is found with an entity in the ontology, this ontological term falls within the candidate classes for classification, otherwise the next term in the ontology must be reviewed.

Chapter 6

Evaluation

In order to verify the goodness and improvements of the classification approach using an ontology, the algorithm, developed in the Python language for this work, was trained on a subset of only 100 instances, given the resulting computational complexity of implementing this approach.
Below are the results obtained with the execution of the classification algorithm, comparing the standard classification with the classification via ontology.

Classifier	Metric	100 Istances without Ontology	100 Istances with Ontology
Multinomial Naive Bayes	**Accuracy**	0.2	0.18
	Precision	0.2	0.18
	Recall	0.2	0.18
	F-measure	0.2	0.18
Bagging	**Accuracy**	0.14	0.14
	Precision	0.14	0.14
	Recall	0.14	0.14
	F-measure	0.14	0.14
K-Nearest Neighboor	**Accuracy**	0.08	0.06
	Precision	0.08	0.06
	Recall	0.08	0.06
	F-measure	0.08	0.06
Support Vector Machine	**Accuracy**	0.24	0.18
	Precision	0.24	0.18
	Recall	0.24	0.18
	F-measure	0.24	0.18
Decision Tree	**Accuracy**	0.08	0.06
	Precision	0.08	0.06
	Recall	0.08	0.06
	F-measure	0.08	0.06
Random Forest	**Accuracy**	0.18	0.12
	Precision	0.18	0.12
	Recall	0.18	0.12
	F-measure	0.18	0.12

Given the small size of the dataset used for the training phase, both approaches perform poorly

in every metric chosen for evaluation. The performance gap can be verified in the Figure 6.1.

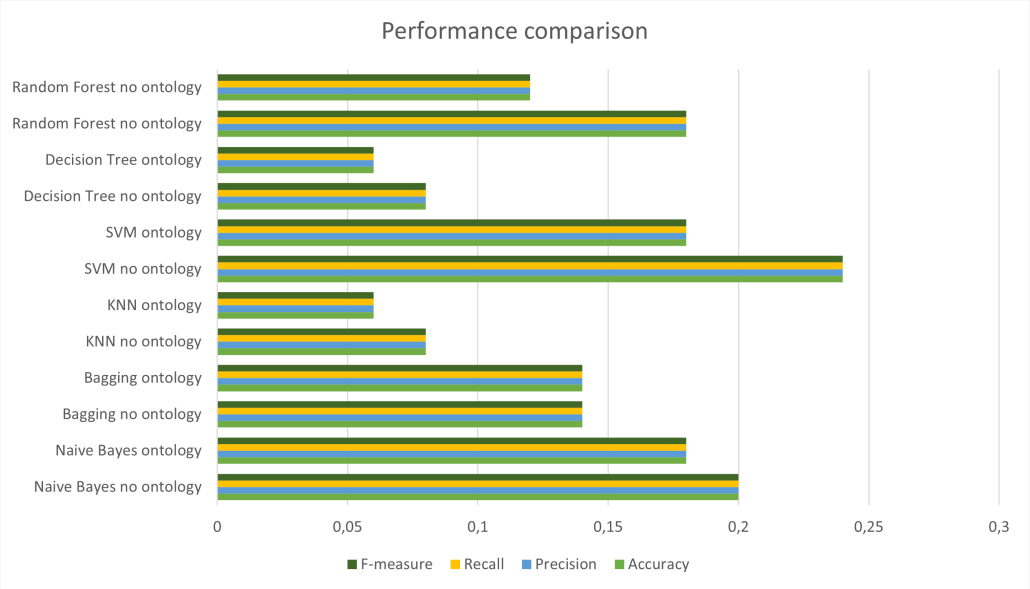

Figure 6.1: Performance metrics comparison between two approaches

These results are best appreciated in the study by Malik, Sonika and Jain, Sarika [12], where the same approach is tested and its performance calculated. It appears from the article that there is a 10% improvement in metrics for 500 test cases and 6% for 100 test cases for decision tree classifier. It is followed by the KNN Classifier, which shows the 5% improvement for both 100 and 500 test cases. Rest other classifiers have shown improvement in metrics of around 1% - 3%.
In the Malik works, we can read that:

The bagging classifier follows next with the values of the parameters being 0.87 each in simple text classification, and 0.89 in ontology based classification. Random Forest Classifier comes next, as it shows the values of the parameters as 0.96 and 0.97 in each classification case. Naïve-Bayes Classifier and the SVM classifier have the same values of all the parameters as 0.98 for simple classification and 1.0 for ontology based text classification. Now we will come to Logistic Regression, which can be labelled as best classifier with the values of metrics as 0.99 for simple Text classification while 1.0 for ontology based text classification.

Given the sparseness of the classification obtained by exploiting a considerably reduced dataset, we wanted to check whether the cause of the deterioration compared to the performance indicated in [12] is only due to the number of instances considered.
In order to do this, we chose to run the algorithm with 500 training instances in order to compare the results obtained with the previously poor results. The differences can be appreciated in the following table, where only one evaluation metric was chosen, however, since the same results were obtained for each of the chosen metrics.

By varying the training dataset from 100 instances to 500 instances, the performance values

Classifier	Metric	100 Istances without Ontology	500 Istances without Ontology	100 Istances with Ontology	500 Istances with Ontology
Multinomial Naive Bayes	**Accuracy**	0.2	0.49	0.18	0.47
Bagging	**Accuracy**	0.14	0.33	0.14	0.38
K-Nearest Neighboor	**Accuracy**	0.08	0.30	0.06	0.41
Support Vector Machine	**Accuracy**	0.24	0.60	0.18	0.68
Decision Tree	**Accuracy**	0.08	0.29	0.06	0.16
Random Forest	**Accuracy**	0.18	0.42	0.12	0.52

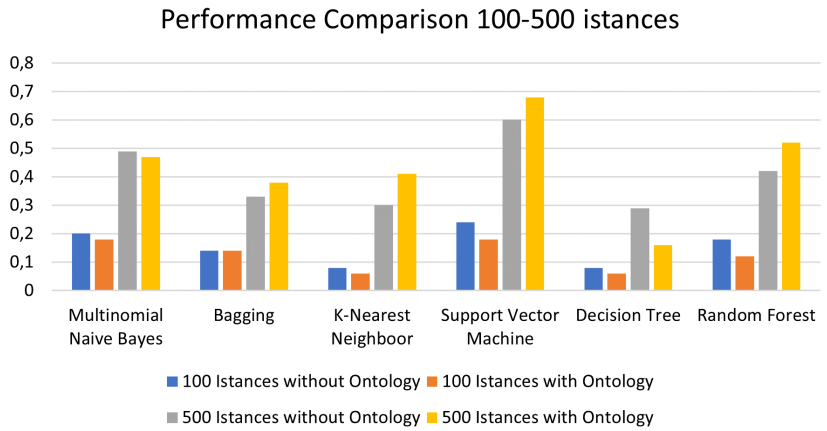

Figure 6.2: Performance evaluation as the dataset size changes

increase considerably, indicating that each classifier has benefited considerably from the increased number of instances, both in its version without ontology and with the addition of the ontology within the process.

In addition, with the higher number of instances, the improvement that the ontology brings to the classification process is more appreciable, especially with regard to the KNN classifier, which has improved its performance in terms of accuracy by approximately 30 per cent.

Strange, however, is the case with the Decision Tree model, which, while benefiting from the increase in instances, does not gain any improvements through the use of ontology. This could be due to the fact that the Decision Tree model still needs more data to realise the benefit of the methodology applied in this work, since it still obtains poor results even in the absence of the ontology, compared to the other models used.

Chapter 7

Conclusions

In this paper, the issue of text classification was addressed, with a focus on a new methodology approach that combines ontologies within the classification process itself, in order to improve the inference performance of the model. The use of an ontology, as described in the literature, opens the way to new combinatorial possibilities between machine learning and the improvement of the data the model has to deal with. The proposal of this work is in fact aimed at improving the data processed by the learning algorithm in order to make the learning phase more precise, thus improving the performance of the model itself.

This advantage is due to the reduction in the number of possible classes to classify and, therefore, the time spent on training. It also showed that accuracy between classifiers is more comparable when using ontologies. Although this study demonstrates the importance and benefits of ontologies, there remains much room for future improvement. Further work is needed to improve the disease database used in this project, as there is no official database of human disease ontologies. Much of the work has been done with limited datasets obtained by converting existing ontologies into datasets. In addition, the code used to match ontologies needs to be optimized after preprocessing the data. It is also possible to move from machine learning to deep learning and build neural networks on this dataset to further improve classification results in the future.

Bibliography

[1] Wikipedia contributors. *Document classification — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Document_classification&oldid=1095886958.

[2] Wikipedia contributors. *Ontology (information science) — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Ontology_(information_science)&oldid=1097810148.

[3] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[5] Wikipedia contributors. *Lexical analysis — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Lexical_analysis&oldid=1104772145.

[6] Wikipedia contributors. *Supervised learning — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Supervised_learning&oldid=1107692217.

[7] Wikipedia contributors. *Unsupervised learning — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=1107188105.

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Wikipedia contributors. *Precision and recall — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=1109375084.

[10] Columbia Department of Biomedical Informatics (DBMI). URL: https://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/.

[11] Lynn M Schriml et al. "The human disease ontology 2022 update". In: *Nucleic acids research* 50.D1 (2022), pp. D1255–D1261.

[12] Sonika Malik and Sarika Jain. “Semantic Ontology-Based Approach to Enhance Text Classification.” In: (2021), pp. 85–98.