UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE BIOLOGIA VEGETAL

Extracting Biomedical Relations from Biomedical Literature

Tânia Sofia Guerreiro Maldonado

Mestrado em Bioinformática e Biologia Computacional
Especialização em Bioinformática

Dissertação orientada por:
Prof. Doutor Francisco José Moreira Couto

2018

Resumo

A ciência, e em especial o ramo biomédico, testemunham hoje um crescimento de conhecimento a uma taxa que clínicos, cientistas e investigadores têm dificuldade em acompanhar. Factos científicos espalhados por diferentes tipos de publicações, a riqueza de menções etiológicas, mecanismos moleculares, pontos anatómicos e outras terminologias biomédicas que não se encontram uniformes ao longo das várias publicações, para além de outros constrangimentos, encorajaram a aplicação de métodos de *text mining* ao processo de revisão sistemática.

Este trabalho pretende testar o impacto positivo que as ferramentas de *text mining* juntamente com vocabulários controlados (enquanto forma de organização de conhecimento, para auxílio num posterior momento de recolha de informação) têm no processo de revisão sistemática, através de um sistema capaz de criar um modelo de classificação cujo treino é baseado num vocabulário controlado (MeSH), que pode ser aplicado a uma panóplia de literatura biomédica.

Para esse propósito, este projeto divide-se em duas tarefas distintas: a criação de um sistema, constituído por uma ferramenta que pesquisa a base de dados PubMed por artigos científicos e os grava de acordo com etiquetas pré-definidas, e outra ferramenta que classifica um conjunto de artigos; e a análise dos resultados obtidos pelo sistema criado, quando aplicado a dois casos práticos diferentes.

O sistema foi avaliado através de uma série de testes, com recurso a *datasets* cuja classificação era conhecida, permitindo a confirmação dos resultados obtidos. Posteriormente, o sistema foi testado com recurso a dois *datasets* independentes, manualmente curados por investigadores cuja área de investigação se relaciona com os dados. Esta forma de avaliação atingiu, por exemplo, resultados de precisão cujos valores oscilam entre os 68% e os 81%.

Os resultados obtidos dão ênfase ao uso das tecnologias e ferramentas de *text mining* em conjunto com vocabulários controlados, como é o caso do MeSH, como forma de criação de pesquisas mais complexas e dinâmicas que permitam melhorar os resultados de problemas de classificação, como são aqueles que este trabalho retrata.

Palavras-chave: prospeção de texto, vocabulários controlados, literatura biomédica, MeSH, classificação binária

Abstract

Science, and the biomedical field especially, is witnessing a growth in knowledge at a rate at which clinicians and researchers struggle to keep up with. Scientific evidence spread across multiple types of scientific publications, the richness of mentions of etiology, molecular mechanisms, anatomical sites, as well as other biomedical terminology that is not uniform across different writings, among other constraints, have encouraged the application of text mining methods in the systematic reviewing process.

This work aims to test the positive impact that text mining tools together with controlled vocabularies (as a way of organizing knowledge to aid, at a later time, to collect information) have on the systematic reviewing process, through a system capable of creating a classification model which training is based on a controlled vocabulary (MeSH) that can be applied to a variety of biomedical literature.

For that purpose, this project was divided into two distinct tasks: the creation a system, consisting of a tool that searches the PubMed search engine for scientific articles and saves them according to pre-defined labels, and another tool that classifies a set of articles; and the analysis of the results obtained by the created system when applied to two different practical cases.

The system was evaluated through a series of tests, using datasets whose classification results were previously known, allowing the confirmation of the obtained results. Afterwards, the system was tested by using two independently-created datasets which were manually curated by researchers working in the field of study. This last form of evaluation achieved, for example, precision scores as low as 68%, and as high as 81%.

The results obtained emphasize the use of text mining tools, along with controlled vocabularies, such as MeSH, as a way to create more complex and comprehensive queries to improve the performance scores of classification problems, with which the theme of this work relates.

Keywords: text mining, systematic review, controlled vocabularies, biomedical literature, MeSH, binary classification

Acknowledgments

To Prof. Dr. Francisco Couto, my advisor, for the guidance and wise words along this path.

To FCT and LASIGE, for support through funding of the *Programa Estratégico da Unidade de I&D LASIGE – Laboratório de Sistemas Informáticos de Grande-Escala* project, with ref. UID/CEC/00408/2013.

To André Lamúrias, whose help was fundamental when anything else seemed to fail.

To Melinda Noronha, Jenni Moore, and Jan Nordvik, for the kindly provided data and the different perspectives provided to the project.

Last but not least, to my family. For all the support along these years, through rough times and moments of joy, and for making me who I am today.

“Para ser grande, sê inteiro: nada
Teu exagera ou exclui.

Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.”

Ricardo Reis, in "Odes"
Heteronym of Fernando Pessoa

Index

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
AUROC	Area Under the Receiver Operator Characteristic
CS	Citation Screening
CV	Cross Validation
EL	Entity Linking
FDA	Food and Drug Administration
FN	False Negative
FP	False Positive
FPR	False Positive Rate
IDF	Inverse Document Frequency
IM	*Index Medicus*
IR	Information Retrieval
k-NN	k-Nearest Neighbors
MAP	*Maximum A Posteriori*
MeSH	Medical Subject Headings
ML	Machine Learning
NCBI	National Center for Biotechnology Information
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PICOS	Participants, Interventions, Comparators, Outcomes and Study design
PR	Precision-Recall
RE	Relation Extraction
ROC	Receiver Operator Characteristic
SCR	Supplementary Chemical Records
SR	Systematic Review
SVC	Support Vector Classification
SVM	Support Vector Machines
TBS	Token Boundary Symbol
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TM	Text Mining
TN	True Negative
TP	True Positive
TPR	True Positive Rate

Section 1

Introduction

Science is currently witnessing the fast pace at which knowledge grows, especially in the biomedical field.

One of the first organizations to index medical literature was the US National Library of Medicine (NLM), in 1879: the *Index Medicus* (IM) was a comprehensive bibliographic index of life science and biomedical science information, that would in 1996 become the MEDLINE database.

The US Food and Drug Administration (FDA) introduced in 1962 a regulatory framework that required proof of the efficiency of new drugs [1], and other countries followed the practice. This led to an inevitable rise in the number of randomized controlled trials (i.e., a study in which the participants are assigned by chance to separate groups, according to the National Cancer Institute), and at the same time, the overall rise in the number of scientific articles, many providing evidence base for these trials. In 1966, the NLM had indexed 165,255 articles for *Index Medicus;* in 1985, the number of articles was 73% higher, with a total of 286,469 articles indexed [2]. By 2006, the index had grown to nearly 10 million references [3] that would cover areas such as medicine, nursing, pharmacy, dentistry, veterinary medicine, and healthcare. As of 2017, PubMed (a search engine that primarily accesses the MEDLINE database) contains more than 27 million citations for biomedical literature.

As the number of clinical trials raised, so did the science of reviewing trials, which aim to make sense of multiple studies. According to Bastian [3], there are now 75 new trials and 11 new systematic reviews (SR) of trials per day, haven't yet reached a plateau in growth.

Clinicians and researchers are required to keep up with published scientific studies and use them in their field of work. However, with the massive amount of data that the all-new high-throughput molecular biology techniques and studies now produce, as well as the increasingly widespread adoption of health information systems that store clinical data, evidence-based science is increasingly becoming a more laborious task.

Problem

Finding the best scientific evidence that applies to a given problem is becoming exceedingly difficult due to the exponential growth of biomedical publications, which considers several types of publications such as:

(i) scientific publications,

(ii) patents,

(iii) grey literature (conference reports, abstracts, dissertations, and preprints), and

(iv) a plethora of regulatory, market, financial, and patent intelligence tools.

Scientific journals, the type of publication most widely used, tend to share a general arrangement (Title, Abstract, Introduction, Materials and Methods, Experiments, Results, Discussion, and Summary and Conclusion sections) although with considerable variability across publishers and themes.

Another obstacle lies in the fact that biomedical literature is plentiful in mentions of etiology, molecular mechanisms, clinical conditions, anatomical sites, medications, and procedures. Even though the language used for scientific discussion is formal, the names of the biomedical entities may not be uniform across different writings.

This plenitude of different terminologies motivates the application of text mining (TM) methods to enable efficient indexing and determination of similarities between the search terms in a given search engine and the retrieved document. Nonetheless, TM has been applied successfully to biomedical documents, for example, to identify protein-protein interactions [4] and associations between drugs [5].

More than recognizing entities within a given set of documents, it is crucial to recognize the search terms as a biomedical term (or set of terms) during the SR process, providing researchers with better tools to systematic review the existing literature. A common strategy involves linking text to a controlled vocabulary.

Objectives

The main objective of this work is to test the hypothesis that TM tools and controlled vocabularies have a positive impact on the systematic reviewing process, either from an aspect of time reduction or regarding performance (i.e., if a given article is relevant to the study or not).

For the accomplishment of this objective, it will be developed a system capable of creating a classification model which training is based on a controlled vocabulary (Medical Subject Headings – MeSH) that can be applied on a variety of biomedical literature.

This will optimistically provide researchers with a semi-automated systematic reviewing tool that aids them in keeping up with scientific studies, regarding the amount of time saved in research, as well as providing better support for decision-making.

The work described in this dissertation comprises two distinct tasks:

(i) the creation of a system consisting of a tool that searches the PubMed search engine for scientific articles and saves them according to pre-defined labels, and another tool that classifies a set of articles;

(ii) the analysis of the results obtained by the created system when applied to two different practical cases.

Results

The system was evaluated initially through a series of tests, using datasets whose classification results were previously known, allowing the confirmation of the obtained results. Afterwards, the system was tested by using two independently-created datasets which were manually curated by researchers working in the field of study. This last form of evaluation achieved, for example, precision scores as low as 68%, and as high as 81% (average score between two classes, on the Humanin dataset), depending on the controlled vocabulary terms used to train the system.

Contributions

The main contribution of this work is a system capable of creating a classification model in which training is based on a controlled vocabulary (MeSH) that can be applied to a variety of biomedical literature[1].

Document Structure

The following sections are organized as follows:

- **Section 2** focuses on all the work done by third-party entities, i.e., it explains the main concepts applied in this research, presents an overview of the state-of-the-art tools in the area, and showcases the resources that will be further applied;
- **Section 3** presents all the work developed for this thesis, including the system developed, the methodology followed and the datasets used;
- **Section 4** demonstrates the results achieved in each study case, and ends with a discussion of all the results obtained;
- **Section 5** presents the conclusions achieved by this work, its limitations, some suggestions for future work, and finishes with some final remarks.

[1] Available at https://github.com/tanmald/MeSH_ifier.git

Section 2

Concepts and Related Work

This section is dedicated to describing some concepts necessary to contextualize this project, namely a description of systematic reviews, text mining, and controlled vocabularies, as well as presenting some related work.

2.1. Systematic Reviews

Systematic reviews were invented as a means to enable clinicians to use evidence-based medicine, to support clinical decisions [6]. SR identify, assess, synthesize, and interpret multiple published and unpublished studies in a given topic, improving decision-making for a variety of stakeholders [7], while also allow identifying research challenges to develop new research ideas.

The systematic reviewing process is conducted through a robust but slow and human-intensive process. According to Jonnalagadda et al. [8], a SR process includes seven steps:

1. Definition of the review question and development of criteria for including studies;
2. Search for studies addressing the review question;
3. Selection of studies that meet the criteria for inclusion in the review – citation screening (CS);
4. Extraction of data from included studies;
5. Assessment of the risk of bias in the included studies, by appraising them critically;
6. Where appropriate, an analysis of the included data by undertaking meta-analyses should be made;
7. Address reporting biases.

For reviews to be systematic, the search task has to ensure relevant literature is retrieved as much as possible, even at the cost of retrieving up to tens of thousands of irrelevant documents. It also involves searching multiple databases. Therefore, reviewers require specific knowledge of dozens of literary and non-literary databases, each with its own search engine, metadata, and vocabulary [6].

Given the amount of time it takes to filter out the immense quantity of research that will *not* be covered, a SR can take a considerable amount of time to complete. This is often a problem, since decision-making needs to happen quite fast, and there is not always the opportunity for a review to be concluded, even if it leads to a better decision.

There are several possible ways to reduce screening workload. As suggested by O'Mara-Eves et al. [9], these may be summed as follows:

- reducing the number of items that need to be screened manually;
- reducing the number of experts needed to screen the items;

- increasing the rate (or speed) of screening;
- improving the workflow.

To reduce the workload, there are ongoing efforts to automate part or all of the stages of the SR process. One approach is the application of Machine Learning (ML) techniques using TM to automate the CS (also called study selection) stage. Since the ML prediction performance is generally on the same level as the human prediction performance, using a ML-based system will lead to significant workload reduction for the human experts involved in the systematic review process [9].

2.2. Text Mining

Tan [10] described TM as “the process of extracting interesting and non-trivial patterns or knowledge from unstructured text documents.” According to Hotho [11], TM is a multi-disciplinary field in computer science that relies on information retrieval, machine learning, statistics, computational language, and data mining.

Research in this area is still in a state of significant flux, indicated by the sometimes confusing use of terms. Hotho et al. [11], for instance, presented different TM definitions, driven by the specific perspective of the area. The first approach considered that TM essentially corresponds to information retrieval (IR); a second strategy referred to TM as the application of algorithms and methods from machine learning and statistics to texts, aiming to find useful patterns.

Regarding biomedical TM, to name a few of the most typical tasks, one can point out:

- Information Retrieval (IR): to rank or classify articles for topics of relevance,
- Named Entity Recognition (NER): detect a variety of different types of bioentity mentions,
- Entity Linking (EL): index or link documents to terms from controlled vocabularies or bio-ontologies, and
- Relations Extraction (RE): extract binary relationships between bioentities, in particular, protein or gene relations, like protein−protein interactions.

Despite the differences in focus and scope of the several biomedical branches, end users have mutual information demands: from finding papers of relevance (IR) to the assignment of predefined classes to text documents (formally known as classification).

The tasks of TM on which this work mainly focuses on are IR and classification, and therefore those will be described in the next sub-sections. A small description of other tasks, not addressed in this work but also relevant to the biomedical domain, will also be presented.

2.2.1. Information Retrieval

The practical pursuit of computerized information retrieval began in the late 1940s; the term

information retrieval was later used for the first time by Calvin Mooers, in 1950 [12].

As defined by Manning et al. [12] “IR is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).” The term “unstructured” mentions data that does not have clear, semantically explicit structure, that is easy for a computer to understand. It is the opposite of structured data, which the better example is a relational database.

In other words, IR is a task of TM that deals with automatically finding relevant texts from large datasets of unstructured text, where manual methods would typically be infeasible [13].

2.2.1.1. Natural Language Processing Techniques

In most of the cases, the information demand concerns human language texts. Natural language processing (NLP) deals with the interactions between computers and human (natural) languages, particularly, with parsing the input text into a machine-readable form.

The following NLP techniques are some of the most commonly used in text mining systems, and they are also broadly applied in the biomedical domain:

Sentence Splitting

A low-level text processing step that consists of separating written text into individual sentences [14]. Follows simple heuristic rules, for example, a space followed by a capital letter should be separated [15]. Some exceptions could be “*Dr. Xxx*” or “*e.g., YYY*.”

Tokenization

Given a character sequence and a defined document unit, tokenization is the task of cutting it into smaller pieces, called tokens [12]. It is usually the first step in a text processing system, and if wrongly implemented, can lead to a poor-performing system [16].

Although these tokens are usually related to single words, they may also consist of numbers, symbols or even phrases. It has been observed that in biomedical documents, symbols that usually correspond to token boundary symbols (TBS), such as “+,” “/” and “%,” do not always denote correct boundary elements.

A tokenization parser is used to retrieve these tokens from the text, splitting the input based on a set of predefined rules. The output of various tokenizers can be significantly different, for instance, depending on how characters such as hyphens are handled [14], [17]. Two examples of systems

developed specially for text written in the English language are the Stanford Tokenizer[2] and Banner[3].

Stemming & Lemmatization

The tokens are usually normalized before being added to a given term list; that is, a linguistic pre-processing step is carried out to generate a modified token representing the canonical form of the corresponding term [14]. Typically, this step refers to either stemming or lemmatization. Both aim to reduce words to their common base form: for instance, “am,” “are” and “is” would become “be”; “car,” “cars,” “car’s” and “cars” would become “car.”

The difference between both techniques is that stemming usually refers to a heuristic process that slices the ends of words, hoping to achieve this goal correctly most of the time. Lemmatization, on the other hand, attempts to perform a vocabulary correctly and morphological analysis of words, typically aiming to remove inflectional endings only and to return the dictionary form of a word (known as the lemma). However, to achieve this, the word form must be known, i.e., the part of speech of every word in the text document has to be assigned. Since this tagging process is usually very time-consuming and error-prone, stemming methods are applied alternatively [11].

Porter’s stemming algorithm[4] has been shown to be empirically very effective [12]. It is a process for removing the commoner morphological and inflexional endings from words in English [18].

The BioLemmatizer[5] is a domain-specific lemmatization tool for the morphological analysis of biomedical literature, achieving very high-performance scores when evaluated against a gold standard of manually labeled biomedical full-text articles [14], [19].

Machine Learning

One approach that has increasingly become the method of choice for many text classification tasks is Machine Learning. ML is a field of computer science which applies statistical techniques so that computer systems can "learn" (i.e., progressively improving its performance on a specific task) with data, without being explicitly programmed for it [20].

Regarding the classification problem, and given a set of classes, the user seeks to determine which class(es) a given document belongs to. More formally, the classification problem is defined

---

2 https://nlp.stanford.edu/software/tokenizer.shtml

3 https://github.com/oaqa/banner/blob/master/src/main/java/banner/tokenization/Tokenizer.java

4 https://tartarus.org/martin/PorterStemmer/index.html

5 http://biolemmatizer.sourceforge.net

as follows: having a training set $D\langle d_i\rangle, i = 1,2,\dots,n$ of documents, such that each document d_i is labeled with a label $C = \{c_1, c_2, \dots, c_j\}$, the task is to find a classification model (a **classifier**) f where

$$f: D \to Cf(d) = c \tag{2.1}$$

Which can assign the correct class label to a new document d (test instance) [21].

There are two main ML categories: supervised, and unsupervised learning. For supervised ML techniques to work well, manually annotated corpora are required as a training set. A statistical model/learning algorithm is "fed" with the training set to learn from it, and subsequently applied to assign labels to previously unseen data. Regarding unsupervised learning, no labels are given to the learning algorithm, and these are typically based on clustering algorithms.

Commonly used annotated corpora in the biomedical domain are the GENIA[6] and the PennBioIE[7] corpora, achieving very high-performance scores [14].

Regarding unsupervised learning, there are several learning algorithms worth emphasising.

i. Multinomial Naïve Bayes

Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naïve" assumption of independence between every pair of features.

The Bayes classifier is a hybrid parameter probability model, that states the following relationship:

$$P(c_j|D) = \frac{P(c_j)P(D|c_j)}{P(D)} \tag{2.2}$$

Where $P(c_j)$ is prior information of the appearing probability of class c_j, $P(D)$ is the information from observations (which is the knowledge from the text itself to be classified), and $P(D|c_j)$ is the distribution probability of document D in classes space [22].

Regarding text classification, the goal is to find the best class for the document (Manning et al., 2009). The best class in Naïve Bayes classification is the most likely, or *maximum a posteriori* (MAP), class c_{map}:

$$c_{map} = arg_j max P(c_j|D) = arg_j max P(c_j) \prod_i P(D_i|c_j) \tag{2.3}$$

Naïve Bayes classifiers work quite well in many real-world situations, namely document

6 http://www.nactem.ac.uk/aNT/genia.html

7 https://catalog.ldc.upenn.edu/LDC2008T21

classification and spam filtering [24], although they require a small amount of training data to estimate the necessary parameters.

The different naïve Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(c_j|D)$. Until this point, nothing was said about the distribution of each feature. One disadvantage of the Naive Bayes is that it makes a very strong assumption on the shape of the data distribution, i.e. that any two features are independent given the output class. As for the multinomial naïve Bayes, it acknowledges that each $P(c_j|D)$ is a multinomial distribution, rather than any other distribution, and is one of the two classic naïve Bayes variants used in text classification [25].

ii. K-Nearest Neighbors

Neighbors-based classification is a type of instance-based or non-generalizing learning, which does not attempt to construct a general internal model, but solely stores instances of the training data [26].

One of the neighbors-based classifiers is the *k*-nearest neighbors (*k*-NN). It is a non-parametric (i.e., not based solely on parameterized[8] families of probability distributions) method used for classification and regression. In any case, the input consists of the k closest training examples in the feature space. Within the case of classification, the output is a class association. The principle behind *k*-NN is that an object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbors.

In ML, the training examples are vectors in a multidimensional feature space, each containing a class label. During its training phase, the algorithm stores the feature vectors and class labels of the training samples. In the classification phase, k is a typically small, positive user-defined constant, and an unlabelled vector (either a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that point. If $k = 1$, then the object is simply assigned to the class of that single nearest neighbour [27]. In binary (two class) classification problems, it is helpful to choose k to be an odd number, as this avoids tied votes.

By default, *k*-NN employs the Euclidean distance, which can be calculated with the following equation:

$$D(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \quad (2.4)$$

where p and q are subjects to be compared with n characteristics [28].

---

[8] Common examples of parameters are the mean and variance.

iii. Decision Trees

Tree models can be employed to solve almost any machine learning task, including classification, ranking, and probability estimation, regression and clustering [29]. In supervised learning, classification trees (common name for when a decision tree is used for classification tasks) are used to classify an instance into a predefined set of classes based on their attribute values, i.e., by learning simple decision rules inferred from the data [30].

Decision trees consist of nodes that form a Rooted Tree, i.e., a tree with a node called a “root” that has no incoming edges. All the remaining nodes have exactly one incoming edge. A node with outgoing edges is referred to as an “internal” or a “test” node. All other nodes are called “leaves.”

Each internal node of the tree divides the instance space into two or more sub-spaces, according to a particular discrete function of the input attributes values. The simplest and most frequent case is the one where each considers a single attribute, i.e., the instance space is partitioned according to the value of the attribute. For numeric attributes, a range is considered. Thus, each leaf is assigned to one class representing the most appropriate target value [30].

Figure 2.1 presents an example of a decision tree that predicts whether or not a potential customer will answer to a direct mailing. Rounded triangles represent the internal nodes (with blue background), whereas rectangles denote the leaves. Each internal node may grow two or more branches. Each node corresponds to a particular characteristic, and the branches correspond with a range of values, which must be mutually exclusive and complete. These two properties of disjointness and completeness are essential to ensure that each data instance is mapped to one instance.

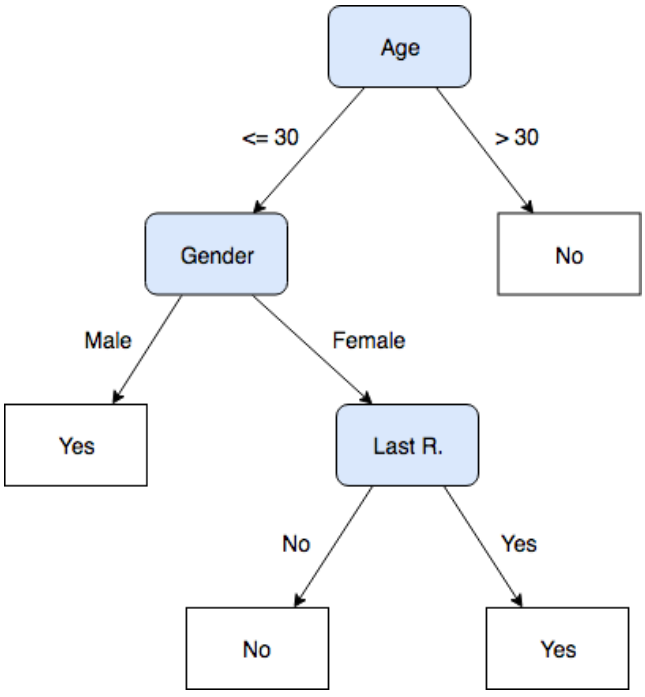

Figure 2.1 - Decision tree presenting response to direct mailing (adapted from [30])

iv. Random Forest

Random forests (also known as random decision forests) are an ensemble learning method (i.e., that use multiple learning algorithms to obtain a better predictive performance than a learning algorithm would alone) for classification, regression, and other ML tasks [31].

This method works by constructing several decision trees (hence the “forest” denomination) at training time. Each tree in the ensemble is built by taking a sample drawn, with replacement, from the training set. In addition, when splitting a node during the construction of the tree, the chosen split is the best among a random subset of the features, instead of the best among all features.

The random forest method is different from linear classifiers[9] since the ensemble has a decision boundary that can’t be learned by a single base classifier. Therefore, the random forest can be classified as an algorithm that implements an alternative training algorithm for tree models. The practical result is that the bias[10] of the forest typically slightly increases (concerning the bias of a single non-random tree). Nevertheless, due to averaging, its variance[11] also decreases, which usually more than compensates for the increase in bias, hence yielding an overall better model [26], [29].

v. Logistic Regression

Logistic regression is a linear classifier whose probability estimates have been logistically calibrated[12], i.e., calibration is an integral part of the training algorithm, rather than a post-processing step.

The output of this algorithm is a binary variable, where a unit change in the input multiplies the odds of the two possible outputs by a constant factor. The two possible output values are often labelled as "0" and "1", which represent outcomes such as correct/incorrect, for example. The logistic model generalises easily to multiple inputs, where the log-odds are linear in all the inputs (with one parameter per input). With some modification, this algorithm can also be applied to categorical outputs with more than two values, modelled by multinomial logistic regressions, or by ordinal logistic regression if the multiple categories are ordered [32], [33].

Logistic regression models the decision boundary directly. That is, if the classes are overlapping, then the algorithm will tend to locate the decision boundary in an area where classes are

---

9 A linear classifier makes a classification decision based on the value of a linear combination of the object’s characteristics.

10 The bias of an estimator is its average error for different training sets.

11 The variance of an estimator indicates how sensitive it is to varying training sets.

12 Calibration is a procedure in statistics to determine class membership probabilities which assess the uncertainty of a given new observation belonging to each of the already established classes.

maximally overlapping, regardless of the 'shapes' of the samples of each class. This results in decision boundaries that are noticeably different from those learned by other probabilistic models, like Naïve Bayes [29].

vi. Support Vector Machines

Linearly separable data admits infinitely many decision boundaries that separate the classes, some of which are better than others. For a given training set and decision boundary, the training examples nearest to the decision boundary (on both sides of it) are called support vectors. Thus, the decision boundary of a support vector machine (SVM) is defined as a linear combination of the support vectors [29]. In supervised learning, an SVM algorithm will build a model that assigns new examples to one category (out of two), making it a non-probabilistic binary linear classifier.

Support vector classification (SVC) and NuSVC are algorithms capable of performing multi-class classification on a given dataset. They both are extensions of the SVM algorithm. These are similar methods but accept slightly different sets of parameters and have different mathematical formulations. Both methods implement the "one-vs-one" approach for multi-class classification [34]. If $n_{class}$is the number of classes, then $n_{class} * (n_{class} - 1)/2$ classifiers are constructed and each one trains data from two classes.

2.2.1.2. Performance Assessment

To evaluate the effectiveness of an IR system (the quality of its results), we can apply two popular evaluation metrics:

- **Precision** (p, or positive predictive value) is the percentage of correctly labeled positive results over all results, i.e., *how many of the selected items are correct*;
- **Recall** (r, also sometimes named coverage, sensitivity, true positive rate, or hit rate) refers to the percentage of correctly labelled positive results over all positive labelled cases, i.e., *how many of the correct items were selected*.

A system with high recall but low precision returns many results, most of which are incorrect when compared to the training labels. The contrary case is a system with high precision but low recall, which returns very few results, but most of its predicted labels are correct when compared to the training ones. An ideal system is the one that returns many results, all of which labelled correctly, achieving high precision and recall values.

Precision and recall can be described as a class match problem where the notion of true positive, true negative, false positive and false negative is required.

For a better understanding, table 2.1 shows a confusion matrix that relates each of these measures. The concepts presented are a result of the relation between the predicted class (the

one assigned in the process) and the golden class (the correct class/assignment).

Table 2.1 - Confusion matrix with evaluation measures

Predicted class	Golden class	
	Positive	**Negative**
Positive	**True positive (TP)**	False positive (FP)
Negative	False negative (FN)	**True negative (TN)**

The stated concepts can be described as follows:

- **True Positive**: If the identified class is correctly labelled, i.e., is present in the golden class.
- **True Negative**: If the class is not present in the golden file, and the system, correctly, did not identify it.
- **False Positive**: cases wrongly misclassified as positive (type I errors, incorrect cases), i.e., the identified class is not present in the golden file;
- **False Negative**: cases missed or incorrectly rejected by the system (type II errors).

The figure 2.2, presented below, may help the comprehension of these concepts.

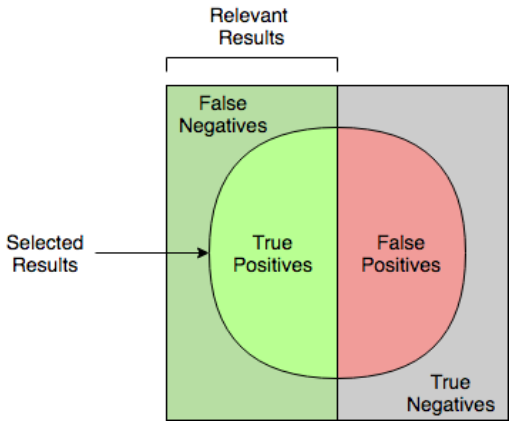

Figure 2.2 - Diagram for comprehension of precision and recall concepts

Based on these concepts, one can see the measures previously described as follows:

$$p = \frac{TP}{TP + FP} \quad (2.5)$$

$$r = \frac{TP}{TP + FN} \quad (2.6)$$

Precision and recall are often combined into a single measure, the **F-score** (f, also F_1-score or F-measure), which is the harmonic mean of precision and recall [35]. F-score reaches its best value at 1 (perfect precision and recall) and worst at 0, and can be represented as follows:

$$f = \frac{2 * p * r}{p + r} \quad (2.7)$$

There are other metrics to consider. **Accuracy** (a), for instance, is the fraction of correctly labelled (positive and negative) results over all results. Research in ML has put aside exhibiting accuracy results when performing an empirical validation of new algorithms. The reason for this is that accuracy assumes equal misclassification costs for false positive and false negative errors. This assumption is problematic, because for most real-world problems one type of classification error is much more expensive than another. For example, in fraud detection, the cost of missing a case of fraud is quite different from the cost of a false alarm [36].

Accuracy can be written following the same line of thoughts as precision and recall:

$$a = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.8)$$

One recommended metric when evaluating binary decision problems is **Receiver Operator Characteristic** (ROC) **curves**, which show how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples [37]. An example of a ROC curve and how it should be interpreted is presented in the annex A, figure 1.

However, ROC curves can present an overly optimistic view of an algorithm's performance if there is a significant skew in the class distribution. This can be addressed using **Precision-Recall** (PR) **curves**. An example of a PR curve can be seen in subsection 2.6.3.5.

A precision-recall curve shows the trade-off between precision and recall for different thresholds. A high area under the curve denotes both high recall and high precision, where high precision represents a low false positive rate, and high recall a low false negative rate. High scores for both measures show that the classifier is retrieving accurate results (i.e., high precision), as well as a majority of all positive results (i.e., high recall).

The main difference between ROC space and PR space is the visual representation of the curves. In ROC space, the False Positive Rate (FPR) is plotted on the x-axis and the True Positive Rate (TPR) on the y-axis. In the PR space, the x-axis plots Recall, and the y-axis plots Precision. The goal in ROC space is to be in the upper left-hand corner; in PR space, the goal is to be in the upper-right-hand corner [38].

2.2.1.3. Cross-Validation

Cross-validation (CV) [39] is a widely used method by the machine learning community since it

provides a simple and effective method for both model selection and performance evaluation.

Ideally, there would be three different approaches to CV [40]:

1. In the simplest scenario, the user would collect one dataset and train the model via cross-validation to create the *best* model possible. Then, it would collect another utterly independent dataset and test it in the previously created model. However, this scenario is the most infrequent (given time, cost or most frequently dataset limitations).
2. If the user has a *sufficiently large* dataset, it would want to split the data and leave part of it to the side (i.e., completely untouched during the model training process). This is to simulate it as if it was a completely independent dataset, since a model that would repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on data not yet seen [26]. This event is called *overfitting*. To prevent it, the user would then build the model on the remaining training samples and test the model on the left-out samples.
3. Lastly, if the user is limited to a smaller dataset, it may not be able to ignore part of the data for model building simply. As such, the data is split into *k folds*, validation is performed on every fold (thus, the name *k*-fold cross-validation) and the validation metric would be aggregated across each iteration.

Since datasets are frequently small, *k*-fold cross-validation is the most used cross-validation method. Under it, the data is randomly divided to form *k* separated subsets of approximately equal size. In the i^{th} fold of the cross-validation procedure, the i^{th} subset is used to estimate the generalised performance of a model trained on the remaining $k-1$ subsets. The average of the generalised performance observed over all *k* folds provides an estimate of the generalised performance of a model trained on the entire sample [41].

2.3. Controlled Vocabularies

In several fields of study, controlled vocabularies exist as a way to organise knowledge for subsequent retrieval of information. An example of knowledge classification is taxonomy[13]. The end-user will most likely focus on one or more topic areas that can be summarised by a network of concepts and associations between them. These typically correspond to domain concepts which are found in thesauri and ontologies [42].

An **ontology** can be defined as “an explicit specification of a conceptualization” [43], thereby including representation, formal naming and/or definition of the categories, properties, and relations of the concepts, data, and entities that it covers. More formally, in information science, the word ontology is applied to a set of logical axioms that model a portion of reality [44].

[13] The practice and science of classification.

The main strength in applying ontologies to data from different fields is the ease with which researchers can share information and process data using computers. As such, ontologies describe knowledge in a way that can be understood by humans and machines alike.

Because modelling ontologies are highly resource-consuming (given that are developed and described in logic-based languages like OWL), there is a preference to reuse existing models, like thesauri, as ontologies instead of developing ontologies from scratch. However, as Kless [45] stated, thesauri cannot be considered a less expressive type of ontology. Instead, thesauri and ontologies must be seen as two kinds of models with superficially similar structures. A qualitatively good ontology may not be a good thesaurus, the same way a qualitatively good thesaurus may not be a suitable ontology.

A **thesaurus** seeks to dictate semantic manifestations of metadata[14] in the indexing of content objects[15] [46]. In other words, it assists the assignment of preferred terms to convey semantic metadata associated with the content object, guiding both an indexer and a searcher in the selection of the same ideal term/combination of terms to represent a given subject.

The aim in using thesauri is to minimise semantic ambiguity by ensuring uniformity and consistency in the storage and retrieval of any manifestations of content objects.

2.3.1. Medical Subject Headings (MeSH)

The Medical Subject Headings (MeSH) thesaurus is a controlled vocabulary for the purpose of indexing, cataloguing and searching journal articles and books related to the life sciences [47].

It was first introduced by Frank Rogers, director of the NLM, in 1960 [48], with the NLM's own index catalogue and the subject headings of the Quarterly Cumulative Index Medicus (1940 edition) as precursors. Initially, it was intended to be a dynamic list, with procedures for recommending and examining the need for new headings [49]. Today it is used by MEDLINE/PubMed database and by NLM's catalogue of book holdings.

Many synonyms and closely related concepts are included as entry terms to help users find the most relevant MeSH descriptor for the concept they seek. In NLM's online databases, many search terms are automatically mapped to MeSH descriptors to ease the retrieval of relevant information.

2.3.1.1. MeSH Structure

MeSH possesses three types of records [50]:

---

14 Data/information that provides information about other data.

15 Any item that is to be described.

i. Descriptors

Unit of indexing and retrieval. The MeSH descriptors are organised in 16 categories, from anatomic terms, organisms, diseases, and so on. Each category is further divided into subcategories. Within each subcategory, descriptors are arrayed hierarchically from most general to most specific in up to thirteen hierarchical levels. Because of the branching structure of the hierarchies, these are sometimes referred to as "trees" [51].

Each descriptor is followed by the number that indicates its tree location. For example, “C16.131” stands for “Congenital Abnormalities.”

ii. Qualifiers

Qualifiers offer a convenient means of grouping together citations which are concerned with a particular aspect of a subject. For example, “liver/drug effects” indicates that the article or book is not about the liver in general, but about the effect of drugs on the liver.

There are 81 topical Qualifiers (also known as Subheadings) used for indexing and cataloguing in conjunction with Descriptors.

iii. Supplementary Concept Records (SCRs)

Supplementary Chemical Records (SCRs), also called Supplementary Records, are used to index chemicals, drugs, and other concepts such as rare diseases for MEDLINE.

SCRs are not organised in a tree hierarchy; instead, each SCR is linked to one or more Descriptors by the Heading. They also include an Indexing Information (II) field that is used to refer to other descriptors from related topics. There are more than 230,000 SCR records, with over 505,000 SCR terms.

2.3.1.2. Online Retrieval with MeSH

The MeSH Browser[16], as an interactive Web application for searching and browsing MeSH data, is the primary way of access to MeSH. However, as the MeSH browser only returns terms, these are to be used in databases such as PubMed.

The main method of using MeSH with PubMed is by providing the search engine with terms in MeSH records. To ensure a Pubmed search uses a MeSH term, the query should have the *[mh]* tag, for example, “Asthma [mh].” This query[17] would retrieve every citation indexed with this Descriptor since PubMed automatically searches on narrower Descriptors indented under the main Descriptor in the MeSH Tree Structures.

---

16 https://meshb.nlm.nih.gov/search

17 A query is a request for information from a database.

If the user has no idea what MeSH term or terms have been used in indexing relevant literature, a text word search may be performed first. For example, if a user is interested in "scalp diseases" - a term not in MeSH, they can search this term in PubMed (title and/or abstract). After seeing particularly relevant citations, the user can look at the citation record (MEDLINE format), and find the MH term “Scalp Dermatoses,” that will be the basis of a new query [52].

2.3.1.3. Example

A quick search through the MeSH Browser allows the user to acquaint itself with the functioning of the database.

Taking “brain” as a search term, for instance. After the insertion of the term in the search box, a full report is displayed, as presented in figure 2 (in the annex B).

The first tab, “Details,” immediately shows the MeSH Heading and its tree number(s) in the first two lines, in this case, “brain” and “A08.186.211” respectively. The following lines present the related annotations, scope notes, entry terms, and other notes. The “Qualifiers” tab shows the related entry combination (for example, “chemistry:Brain Chemistry”) and allowable qualifiers (for example, “anatomy & histology (AH)”). The “MeSH Tree Structures” tab shows the location of the term, as well as the parent and child nodes (if available). For the referred term, the hierarchy tree is presented in figure 2.3.

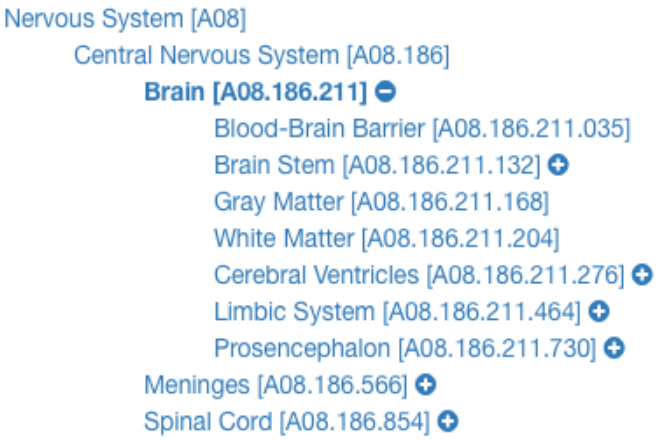

Figure 2.3 - MeSH hierarchy tree for "brain" term

The last tab, “Concepts,” shows the concepts related to the term in question, is this case the only concept is “Brain *Preferred*.”

2.4. Text Mining within Systematic Reviews

Several authors have widely studied the availability and utility of text-mining tools to support systematic reviews over time.

The application of TM techniques to support the citation screening stage of SRs is an emerging research field in computer science, with the first reported publication on the subject in 2005 by Aphinyanaphongs et al. [53]. Their research showed that using machine learning methods it was possible to automatically construct models for retrieving high-quality, content-specific articles in a given time period in internal medicine, that performed better than the 1994 PubMed clinical query filters.

A SR of 26 studies, performed by Pluye et al. [54], reiterates the statement that information-retrieval technology produces a positive impact on physicians regarding decision enhancement, learning, recall, reassurance, and confirmation of a given hypothesis.

In 2015, another SR of 44 papers by O'Mara-Eves et al. [9] pulled together the evidence base for the use of TM for CS. Whilst the authors found that it is difficult to establish any overall conclusions about the best approaches, they also suggested that the (semi)-automation of screening could result in a saving in workload of between 30% and 70%, though sometimes that saving is accompanied by a 95% recall (i.e., the loss of 5% of relevant studies).

Jonnalagadda et al. [8] later referred on their study that the data extraction step is one of the most time-absorbing of the SR process, and that TM techniques, more specifically NLP, may be an essential strategy to reduce the time implicated. Nonetheless, the authors point out that even though most NLP research has focused on reducing the workload for the CS step, biomedical NLP techniques have not been fully exploited to entirely or partially automate the SR process.

A challenge that was pointed by Paynter et al. [55] was that the creation of training datasets, given the comprehensive nature of the TM algorithm, given the comprehensive nature of the research performed, tends to include much more irrelevant than relevant citations, leading to "imbalanced datasets." Olorisade et al. [56] also highlight that the lack of information about the datasets and machine learning algorithms limits the reproducibility of a high amount of published studies.

Even though TM tools are currently being used within several SR organizations for a variety of review processes (e.g., searching, screening abstracts), and the published evidence-base is growing fairly rapidly in extent and levels of evidence, Paynter et al. [57] acknowledge that text mining tools will be increasingly used to support the conduct of systematic reviews, rather than substituting current literature retrieval and information extraction tools. Some significant limitations presented by the authors are that many TM tools rely on *corpora* from PubMed/MEDLINE to train the learning algorithm, which does not represent the entire population of literature relevant for healthcare-related systematic reviews.

2.5. Related Tools

In 2005, Aphinyanaphongs et al. [53] conducted a research where ML methods were used

together with articles cited by the ACP Journal Club as a gold standard for the training of the algorithm. The authors chose this specific gold standard because of its focused quality review, that is highly regarded and uses stable explicit quality criteria.

In most of the studied categories, the data-induced models showed better or comparable precision, recall, and specificity than the pre-existing query filters. These results proved that, following this approach, it is possible to automatically build models for retrieving high-quality, content-specific articles in a given time period that performed better than the 1994 PubMed clinical query filters.

Rathbone et al. [58] evaluated the performance of Abstrackr, a semi-automated online tool for predictive title and abstract screening. The authors used four different SR to train a classifier, and then predict and classify the remaining unscreened citations as relevant or irrelevant. The results showed that the proportion of citations predicted as relevant by Abstrackr was affected by the complexity of the reviews and that the workload saving achieved varied depending on the complexity and size of the reviews. Still, the authors concluded that the tool had the potential to save time and reduce research waste.

Paynter et al. [55] conducted a research which goal was to provide an overview of the use of TM tools as an emerging methodology within some SR processes. This project culminated in a descriptive list of text-mining tools to support SR methods and their evaluation. The authors found two major TM approaches:

1. The first approach assessed word frequency in citations as presented by stand-alone applications, which generate frequency tables from the results set outlining the number of records by text word, controlled vocabulary heading, year, substances, among others. While this approach was used by Balan et al. [59], Kok et al. [60] and Hausner et al. [61] in their studies and applications, other authors used EndNote (a citation management application) to generate word frequency lists.
2. The second approach is automated term extraction. This approach also generates word frequency tables, but many were limited to single word occurrences. Tools such as AntConc[18], Concordance[19], and TerMine[20] extract phrases and combination terms; other applications such as MetaMap[21] and Leximancer[22] add a semantic layer to the process by using tools provided through the NLM's Unified Medical Language System.

18 http://www.laurenceanthony.net/software/antconc/

19 http://www.concordancesoftware.co.uk

20 http://www.nactem.ac.uk/software/termine/

21 https://metamap.nlm.nih.gov

22 https://info.leximancer.com

Even though the tools apply different algorithms, the overall approaches were similar. They start by creating a training set. In addition to that, another corpus representing the general literature (usually created by randomly sampling citations from PubMed) may be presented to the algorithm.

Only “overrepresented” words and phrases in the training set are considered for inclusion in the search strategy. Nonetheless, as noted by Petrova et al. [62] and O’Mara-Eves et al. [63], this approach has inherent problems: not only the reported frequencies for text words do not necessarily reflect the number of abstracts in which a word appears, but the term extraction algorithm also depends on the content of the documents supplied to it by the user/reviewer.

Most of the tools and studies examined by Paynter et al. [55] found benefit in automating term selection for SR, especially those comprising large unfocused topics. For example, in their study, Balan et al [59] concluded that “the benefits of TM are increased speed, quality, and reproducibility of text process, boosted by rapid updates of the results”; Petrova et al. [62] highlights the importance of word frequency analysis, since it “has shown promising results and huge potential in the development of search strategies for identifying publications on health-related values”.

2.6. Resources

This project is built on a wide range of Python packages, namely Biopython, NLTK, and Scikit-learn. The following subsections will describe each of them, relating them to their future role on this work.

2.6.1. Biopython

The Biopython Project [64] is an international association of developers of freely available Python tools for computational molecular biology. Python is an object-oriented, high-level programming language with a simple and easy to learn syntax, which is why it is becoming increasingly popular for scientific computing. Thus, Biopython provides an online resource for modules, scripts, and web links for developers of Python-based software for bioinformatics use and research.

One of Biopython’s functionalities is the access to NCBI’s Entrez databases. Entrez[23] is a data retrieval system that provides users access to NCBI’s databases such as PubMed, GenBank, GEO, among others. Entrez can be accessed from a web browser to enter queries manually, or one can use Biopython’s *Bio.Entrez* module for programmatic access to Entrez, which allows searching PubMed from within a Python script.

After using *Bio.Entrez* to query PubMed, the result will be a Python list containing all of the PubMed IDs of articles related to the given query. If one wishes to get the corresponding Medline records and extract the information from them, it will be necessary to download the Medline

[23] http://www.ncbi.nlm.nih.gov/Entrez

records in the Medline flat-file format and use the *Bio.Medline* module to parse them into Python utilisable data structures.

2.6.2. NLTK

The Natural Language Toolkit, also known as NLTK [65], is an open source library, which includes extensive software, data, and documentation, that can be used to build natural language processing programs in Python. It provides basic classes for representing data relevant to natural language processing, standard interfaces for performing tasks such as syntactic parsing and text classification, and standard implementations for each task that can be combined to solve complex problems.

One of NLTK's functionalities is the processing of raw text. For that, it requires a corpus. The *nltk.corpus* Python package defines a collection of corpus reader classes, which can be used to access the contents of a diverse set of corpora. An example of this is the *CategorizedPlaintextCorpusReader*. It is used to access corpora that contain documents which have been categorised for topic, label, *etc*. In addition to the standard corpus interface, these corpora provide access to the list of categories and the mapping between the documents and their categories.

After accessing the corpus, it is necessary to normalise it. NLTK provides tools to normalize text, from tokenization, the removing of punctuation, or converting text to lowercase, so that the distinction between "The" and "the," for example, is ignored. Another resource NLTK provides is a set of stopwords, that is, high-frequency words like "the," "to" and "also" that one sometimes wants to filter out of a document before further processing. Stopwords usually have little lexical content, and their presence in a text fails to distinguish it from other texts. Often it is still necessary to go further than this, so NLTK offers a way to Stemm and/or Lemmatize the raw text.

2.6.3. Scikit-learn

Scikit-learn [26] is a free machine learning library for Python. It features various classification, regression and clustering algorithms including support vector machines, random forests, k-means and many others.

The Scikit-learn Application Programming Interface (API) is an object-oriented interface centered around the concept of an estimator — broadly any object that can learn from data, be it a classification, regression or clustering algorithm. Each estimator in Scikit-learn has a *fit()* and a *predict()* method:

- The *fit()* method sets the state of the estimator based on the training data. Usually, the data is comprised of a two-dimensional array *X* of shape "(nr. samples, nr. predictors)" that holds the feature matrix, and a one-dimensional array *y* that holds the labels;
- The *predict()* method generates predictions: predicted regression values in the case of

regression, or the corresponding class labels in the case of classification [66].

The main steps of a classification task will be described below, as well as their implementation according to Scikit-learn.

2.6.3.1. Vectorization

To run machine learning algorithms in a corpus or any text document, it is necessary to convert the text into numerical feature vectors. The bag-of-words model [67] (also known as the vector space model) is frequently used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

The problem of just counting the number of words in each document is that it will give more weight to longer documents than shorter documents. To avoid this, term frequency-inverse document frequency (TF-IDF) [68] can be used. TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is the product of two statistics, term frequency (TF) and inverse document frequency (IDF). TF of a word is the frequency of a word (i.e., the number of times it appears) in a document. IDF, on the contrary, reduces the weight of terms that occur very frequently in the document set and, at the same time, increases the weight of terms that occur rarely.

Scikit-learn provides methods to vectorize the data (both the bag-of-words and TF-IDF approaches) through *sklearn.feature_extraction.text*, namely *TfidfVectorizer*. After it has been initialized, the vectorizer works with two methods: *fit_transform()*, and *transform()*. These methods work as follows: to center the data (i.e., make it have zero mean and unit standard error), it is necessary to subtract the mean of the population (μ) and then divide the result by the standard deviation (σ):

$$x' = \frac{x - \mu}{\sigma} \tag{2.9}$$

This procedure is done on the training set of the data. After it, the same transformation has to be applied to test set (e.g., in cross-validation) or to newly obtained examples before forecast. The same two parameters μ and σ that were used to center the training set have to be used. Hence, every sklearn's *fit()* method calculates the parameters (μ and σ) and saves them as an internal object state. Afterward, the *transform()* method is called to apply the transformation to a particular set of examples. The *fit_transform()* method joins these two steps in one and is used for the initial fitting of parameters on the training set x, but it also returns a transformed x'. Internally, it just calls first *fit*() and then *transform*() on the same data.

2.6.3.2. Cross-Validation

Scikit-learn offers several methods to deal with cross-validation through class

sklearn.model_selection, namely *cross_val_predict*. This method works with both the data and the estimator (see 2.6.3.3 for examples of estimators). After splitting the data (both samples and labels) into training and testing sets, it will use the samples and labels of the training set to fit the estimator. Later, this estimator will predict the labels on the test set samples (without using test set labels). This process will be repeated for N times (N is a number defined by the user), each time using different data for training and different data for testing.

2.6.3.3. Classification

Scikit-learn features various classification algorithms based on machine learning. The theory behind these algorithms is explained in 2.2.1.1. Every algorithm has a *fit()* and a *predict()* method, as explained in 2.6.3.

The following classification algorithms take as input two arrays: an array X, sparse or dense, of size $[n_{samples}, n_{features}]$ holding the training samples, and an array y of integer values, size $[n_{samples}]$, holding the class labels for the training samples.

Multinomial Naïve Bayes

MultinomialNB() method, from class *sklearn.naive_bayes*, implements the naïve Bayes algorithm for multinomially distributed data. The multinomial distribution typically requires integer feature (i.e., word vector) counts, however, practically, fractional counts (such as TF-IDF) also work.

K-Nearest Neighbors

From class *sklearn. Neighbors*, the *KNeighborsClassifier()* method implements the k-nearest neighbor's classifier.

Decision Tree

The *DecisionTreeClassifier()* method from class *sklearn.tree* is capable of performing both binary (with labels from range [-1, 1]) and multiclass (with labels from range [0, …, K-1]) classification on a given dataset.

This algorithm can be used to predict the class of the samples after being fitted or, alternatively, the probability of each class (that is, the fraction of training samples of the same class in a leaf).

Random Forest

The *RandomForestClassifier()* method implements the random forest algorithm and can be imported from the *sklearn.ensemble* module.

Logistic Regression

The implementation of logistic regression in Scikit-learn can be accessed from *LogisticRegression()*, imported from *sklearn.linear_model*. This implementation can fit binary, One-vs-Rest, or multinomial logistic regression with optional L2 or L1 regularisation[24].

There are some parameters to this method worth emphasising, namely:

- "C": a positive float with a default value of "1.0". It represents the inverse of regularisation strength, where smaller values specify stronger regularisation;
- "penalty": a string of choice 'l1' or 'l2', with 'l2' as the default value. Used to specify the norm used in the regularisation.

To choose the best parameters for the estimator, an exhaustive search over specified parameter values can be performed using GridSearchCV. This method is detailed in *vii*, below.

Multi-Class Classification

The *sklearn.multiclass* module implements meta-estimators to solve both multiclass and multilabel classification problems, by decomposing them into binary classification problems.

SVC and NuSVC, individually, are handled by Scikit-learn as a One-vs-One strategy, which constructs one classifier per pair of classes. At prediction time, the class which received the most votes is selected. If a tie occurs, (among two classes with an equal number of votes), it selects the class with the highest aggregate classification confidence, by summing over the pair-wise classification confidence levels computed by the underlying binary classifiers.

The *SVC()* and *NuSVC()* methods can be imported from the *sklearn.svm* class.

Grid Search

Estimators may contain parameters that are not directly learned within estimators, also known as hyper-parameters. In Scikit-learn, these are passed as arguments to the constructor of the estimator.

Any parameter provided when constructing an estimator may be optimised by an exhaustive grid search. The grid search provided by *GridSearchCV()*, from class *sklearn.model_selection*, exhaustively generates candidates from a grid of parameter values specified.

GridSearchCV() implements a *fit()* and a *score()* method. It also implements other methods, if they are implemented in the estimator used. The parameters of the estimator used to apply these methods are optimised by cross-validated grid-search over the previously specified parameter

---

[24] Regularisation is the application of a penalty to reduce overfitting the data.

grid. In the end, *GridSearchCV()* returns a list with the estimated best parameter value(s) to choose.

2.6.3.4. Performance Analysis

Class *sklearn.metrics* provides several ways to access the performance of the estimator in Scikit-learn.

If the user wishes to know the accuracy score, the method *accuracy_score()* will return its value. There are two parameters obligatory to fill in this method: *y_true* stands for a label indicator array, containing the correct labels for the test set; *y_pred* is a label indicator array with the predicted labels, as returned by a classifier.

Another way to evaluate the accuracy of a classification is to compute a confusion matrix. By definition [26], a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i, but predicted to be in group j. Therefore, in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

In Scikit-learn, a confusion matrix can be obtained with the *confusion_matrix()* method. As with *accuracy_score()*, it also receives the parameters *y_true* and *y_pred*.

For a more detailed performance analysis of the results, a *classification_report()* method is available. This method builds a text report showing the main classification metrics: precision, recall, f1-score, and support (the number of objects in each class), as well as an average value for all the classes.

Classification_report() takes as input three main parameters: *y_true* and *y_pred* (like *accuracy_score()*), and *target_names*, that may receive either a list of strings, where each is a different class label, or a pointer to a variable containing the class labels (in list of strings format as well). An example of a classification report, as outputted by a Python interpreter, is shown in figure 2.4.

	precision	recall	f1-score	support
class 0	0.67	1.00	0.80	2
class 1	0.00	0.00	0.00	1
class 2	1.00	0.50	0.67	2
avg / total	0.67	0.60	0.59	5

Figure 2.4 - Classification report output example

2.6.3.5. Model Evaluation

Scikit-learn possesses several ways to evaluate the employed models. All text labels must be converted to integers, with *LabelEncoder()* from class *sklearn.preprocessing*, before applying any other function.

Learning Curve

A learning curve presents the user with the validation and training score of an estimator for a set of training samples. It is a way to figure out if and how much a user will benefit from adding more training data and whether the estimator suffers more from a variance or a bias error. If both the validation score and the training score merge into a value that is too low with increasing size of the training set, the user will not benefit much from more training data.

Figure 2.5 shows an example of a learning curve of a naive Bayes classifier, for Scikit-learn "digits" dataset. The training and cross-validation score are both not very good at the end. However, the shape of the curve is representative of more complex datasets [26]: the training score is very high at the beginning and decreases with the increase of training examples, as for the cross-validation score, starts as very low and increases with the increase of training examples. An ideal learning curve would have both scores around the maximum value.

From class *sklearn.model_selection*, the function *learning_curve()* generates the values that are required to plot a learning curve (number of samples used, average scores on training sets and average scores on validation sets).

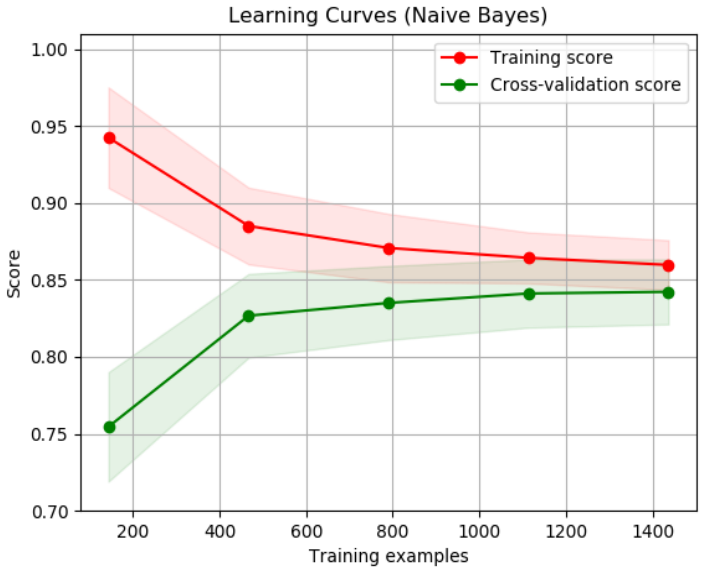

Figure 2.5 - Example of learning curve using Naïve Bayes classifier

ROC Curve

The ROC curve can be computed in Scikit-learn by the *roc_curve()* function. This function requires the true binary value and the target scores, which can either be probability estimates of the

positive class, confidence values, or binary decisions. It is not sensitive to whether the dataset is balanced or imbalanced. An example of a ROC curve is shown in figure 2.6.

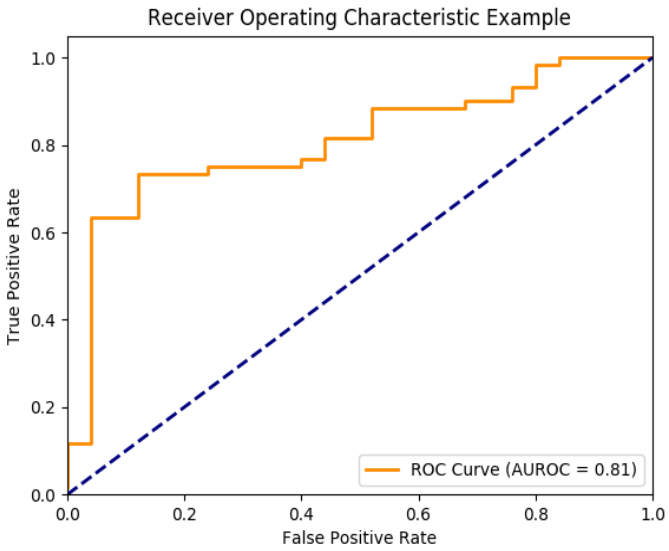

Figure 2.6 - Example of ROC curve

The *roc_auc_score()* function computes the area under the ROC curve, also denoted by AUC or AUROC. AUC can be interpreted as the probability that the classifier will assign a higher score to a randomly chosen positive example, rather than to a randomly chosen negative example [69].

PR Curve

To compute precision-recall pairs for different probability thresholds within the binary classification task, Scikit-learn provides the function *precision_recall_curve()* from module *sklearn.metrics*.

The PR curve is very sensitive to whether the dataset is balanced or imbalanced. An example of a PR curve is shown in figure 2.7.

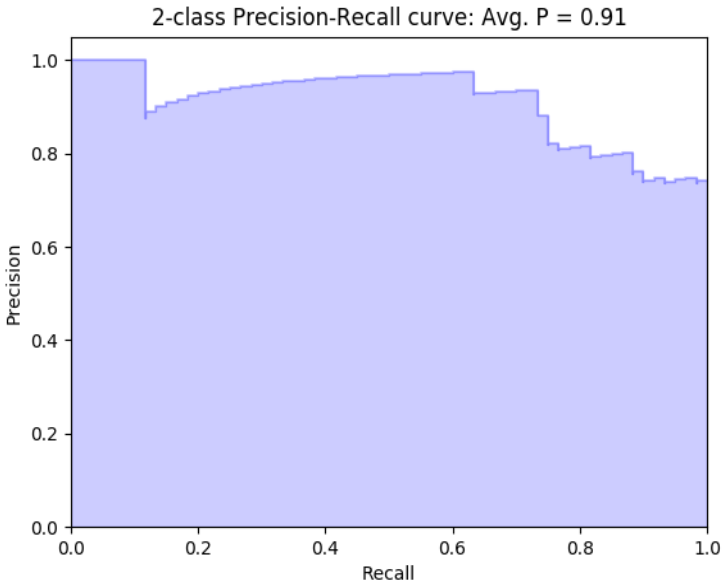

Figure 2.7 - Example of precision-recall curve with average precision of 0.91

Section 3
Developed Work

This section describes all the work developed in order to accomplish the proposed objectives. One of the main goals is the development of a semi-automatic tool for classification of scientific articles, relying on the use of MeSH terms for the enhancement of the performance of the classifier.

First, the methodology followed in this work is presented. An overview of the tool is presented next, where its architecture from a higher level of abstraction its described. Then, a detailed description of its respective components is described, along with the datasets further used by them.

3.1. Methodology

The scheme presented in figure 3.1 represents the overall flow of this work, with research questions and a summary of the employed methodologies for each for the tasks.

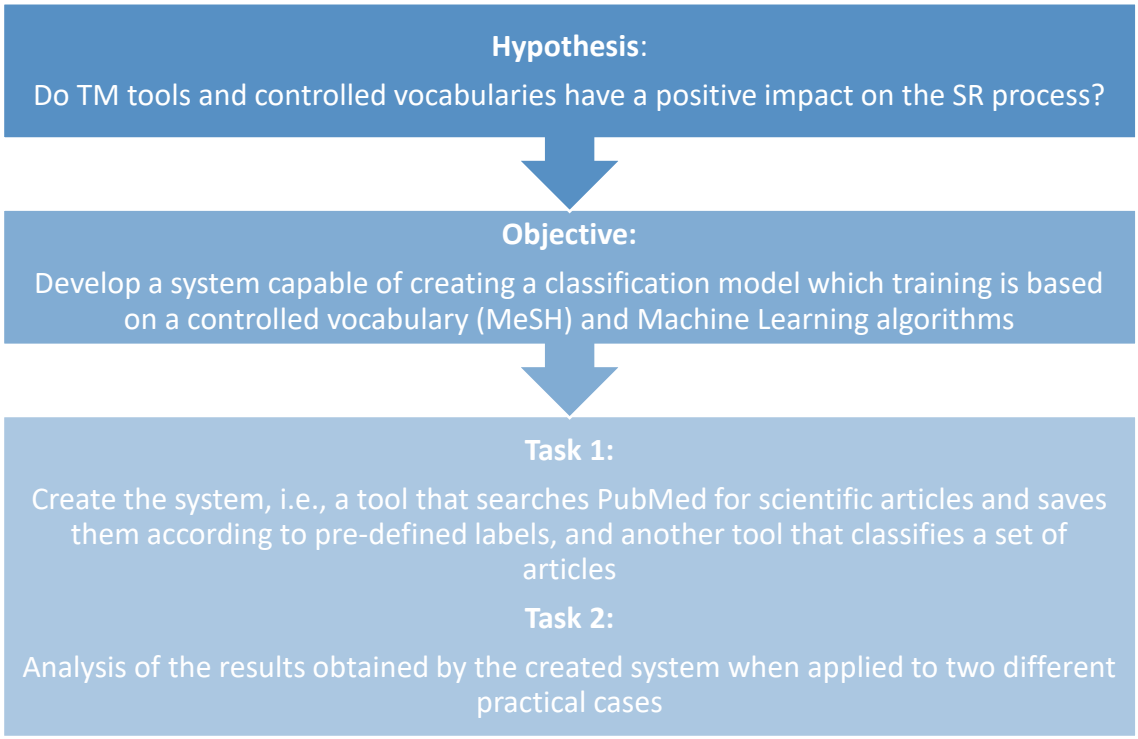

Figure 3.1 - Proposed methodology

3.2. Overview

The system is composed of two major modules: the PubMed search and save, and the classifier modules. Although these modules are responsible for addressing each task independently, they

are part of the same pipeline, presented in figure 3.2.

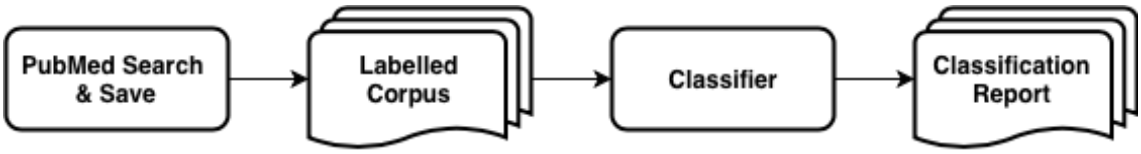

Figure 3.2 - Proposed pipeline

Figure 3.2 represents a higher level of abstraction of the system execution. The system starts with a user-presented query into the PubMed Search & Save script, which outputs a labelled corpus. Two different corpora will be inputted to the Classifier script, which will produce a classification report for the user to analyse.

If there is no need to download a new corpus from PubMed, i.e., if the user already has a set of ".txt" format files containing the title and abstract for a given article, the first two steps of the pipeline may be skipped. For that set of articles to be used as a corpus, they should all be inserted in the same folder and follow the filename scheme of "$label_N.txt$", where "label" is the class label the user desires, "N" is a number (all files must have different numbers) and ".txt" is the file format.

3.3. Script Development

To accomplish the tasks proposed, two scripts were developed: one for searching and saving articles from PubMed according to a given query, and a second one for the classification of a given corpus. The following sub-sections will describe the implementation of each.

3.3.1. PubMed Search & Save

The "PubMed Search and Save" python script was built with the intent of facilitating both the article retrieval and the future usage of the retrieved articles into the classifier script. It is built under BioPython modules, and should be run on Terminal, by accessing the directory where the script is located and then using the command "python pubmed_search_and_save.py."

The script is built in a user-friendly way: after successfully initiating the script, the user only needs to provide the script with a few information for it to run. An example run is showed in figure 3.3, below.

The email is a mandatory field since to make use of NCBI's[25] E-utilities, NCBI requires an email address with each request. The reason for this is because, in case of excessive usage of the E-utilities, NCBI will attempt to contact a user at the email address provided before blocking access to the E-utilities.

[25] The National Center for Biotechnology Information (NCBI) is part of the NLM.

The following question regards the desired query. This field should be answered just like a regular PubMed search.

```
Starting PubMed Search & Save. Please answer a few questions:
What's your email? (NCBI requires you to specify your email address with each request) xxxx@gmail.com
Please enter the desired query: mindfulness[MeSH Terms]
Please enter the desired label for the retrieved files: mindfulness
What's the mininum date for search? (YYYY/MM/DD) 2018/01/01
Finally, how many files do you wish to retrieve? 5


Saving information for 5 out of 5 articles.
Done!
```

Figure 3.3 - “PubMed Search and Save” example run

The “desired label” question is a way of identifying the retrieved articles, not only for the user itself but especially for usage with the classifier script (as it needs that each document from the corpus to have a label). The label should have no spaces – if necessary; the user should instead use the “_” symbol.

The minimum date field is optional – the user may or may not wish to limit the search by a given date. In any case, a date should be provided in the YYYY/MM/DD format, even if it is “0000/01/01”.

The latter question is related to the number of articles desired. The script will try to fetch as many articles as inputted by the user, and the final number of articles retrieved is shown below. If there is any problem with an article, the script will pass to the next one and show a “Saving information for X out of Y articles” information.

All articles are saved to the same directory where the script is located.

3.3.2. Classifier

The “Classifier” python script was built in consideration with the research questions and benefiting from Scikit-learn’s state-of-the-art implementations of many well-known ML algorithms, among other Python modules. It is projected to run on Terminal, by accessing the directory where the script is located and then using the command “python classifier.py.”

The script is built in a user-friendly way: after successfully initiating the script, the user only needs to provide the script with a few information for it to run. An example run is showed in figure 3.4.

For a correct usage of the script, the user is requested to enter the location of both training and test data. It is required that both corpora are located in different paths, to ensure better results.

After specifying the location of the corpora, the user is requested to choose a classification algorithm from the provided list. For it, it should simply enter a number corresponding to the desired algorithm.

Following the selection of the algorithm, the user is asked whether desires to see the plot for learning curves, precision/recall, ROC or confusion matrix, on which it should reply with Y/y for "yes," and N/n for "no." If the answer is "yes" for any option, a graphic will be computed following the examples presented in section 2.6.3.5.

```
Starting the Classifier. First, let's set everything up.
Please specify the location of the training data: /Users/taniamaldonado/PycharmProjects/corpora/humanin/train4
Please specify the location of the test data: /Users/taniamaldonado/PycharmProjects/corpora/humanin/test

Please choose a classification algorithm:
1. Multinomial Naive Bayes
2. K Neighbors
3. Random Forest
4. Decision Trees
5. Logistic Regression

Enter a number to select the algorithm: 1

Do you wish to see any of the above:
Learning curve plot: (Y/N) n
Precision/recall plot: (Y/N) n
ROC curve plot: (Y/N) n
Confusion matrix plot: (Y/N) n

Starting the classifier...

Validation set document classification accuracy = 94.8811700183
Test set document classification accuracy = 75.2941176471
             precision    recall  f1-score   support

nonrelevant       0.61      0.44      0.51        25
   relevant       0.79      0.88      0.83        60

avg / total       0.74      0.75      0.74        85

[[11 14]
 [ 7 53]]

Finished!
```

Figure 3.4 - "Classifier" script example run

After the initial setup of the classifier, a few measures are printed, as well as the full classification report.

The "validation set document classification accuracy" is a percentage, related to the ten-fold cross-validation performed with the training data, and consequent label prediction. It is followed by a "test set document classification accuracy" that uses percentage as well, as an indicator of the accuracy of the classification algorithm predictions.

A more accurate and in-depth classification report is provided in the following lines. It should be read as a table, where the first line contains the column names, i.e., the measures in study ("precision," "recall," "f1-score") and the number of articles in evaluation ("support"). The first and second rows represent each of the data labels (the ones chosen by PubMed Search and Save "desired label" parameter), and the last row is an average measure for each of the classification results and the total number of articles, in the "support" column. Each measure (for precision, recall, and f1-score) is shown as a number from 0 to 1.

The last line provides a simple confusion matrix, where the first line is related to the first label (in

the example, “nonrelevant”) and the latter line to the second label (in the example, “relevant”). For this specific example, the reading would be “for the ‘nonrelevant’ label, there were 11 articles correctly classified and 14 incorrectly classified; for the ‘relevant’ label, there were seven articles incorrectly classified and 53 correctly classified.”

3.3.2.1. Model Evaluation

Before applying the classifier to the datasets in study, a few tests were conducted in order to verify its correct functioning.

A first corpus of 43 articles was generated, with 23 articles belonging to a category with label “breast_cancer” and the remaining 20 to a category with “hd” (from Huntington’s disease) label. The “hd” articles were obtained using the “PubMed Search and Save” script, using the following query: "(huntington disease[MeSH Terms]) NOT breast cancer[MeSH Terms]" and a minimum date parameter of '2015/01/01'; the “breast_cancer” articles were obtained using the following query: "(breast cancer[MeSH Terms]) NOT huntington disease[MeSH Terms]", with a minimum date parameter of ‘2016/01/01’.

The intent with using such different subjects was to ensure that each group of articles had a different bag of words. This can be attested by the figure 3 (in the annex C), showing each bag’s set of words.

After verifying that each corpus had different bags of words, a bigger corpus (with the same queries) with approximately 2000 articles was generated. With this new corpus, a set of tests was made to fine tune the classifier algorithm, aiming to achieve a precision of 80% and F-score of 85%.

With the successful achievement of these target values, a final test was performed: the classifier was trained using the same corpus, but a different test set containing “fake” “hd” label articles (i.e., cancer articles whose label was intentionally replaced to “hd”) was fed to the classifier. The result was a “correct misclassification” of these articles, i.e., the classifier correctly classified the “hd” articles as belonging to the “cancer” category. A confusion matrix that attests these results can be seen in figure 4 (in the annex D).

3.4. Datasets

The application of TM and NLP techniques requires annotated datasets in order to develop and/or evaluate new approaches. These datasets are made up of a corpus of documents, relevant to a specific domain, and its annotations. Since most times these are manually curated by domain experts, they can serve as a gold standard to train, for instance, a ML classifier and evaluate its performance. The downside of manually curated annotations is that they require a defined set of annotation guidelines and availability to annotate the texts [12], [70].

For the development of this work, two different and independently curated datasets have been created and used.

3.4.1. Mindfulness/Fatigue

Ulrichsen et al. [71] developed a systematic review to study the efficiency of mindfulness-based interventions for fatigue across neurological conditions and acquired brain injuries.

Systematic literature searches were conducted in PubMed, Medline, Web of Science, and PsycINFO, using “fatigue” and “mindfulness” as query keywords. A total of four studies (out of 372) were retained for meta-analysis. Figure 3.5 summarises the search and study selection processes.

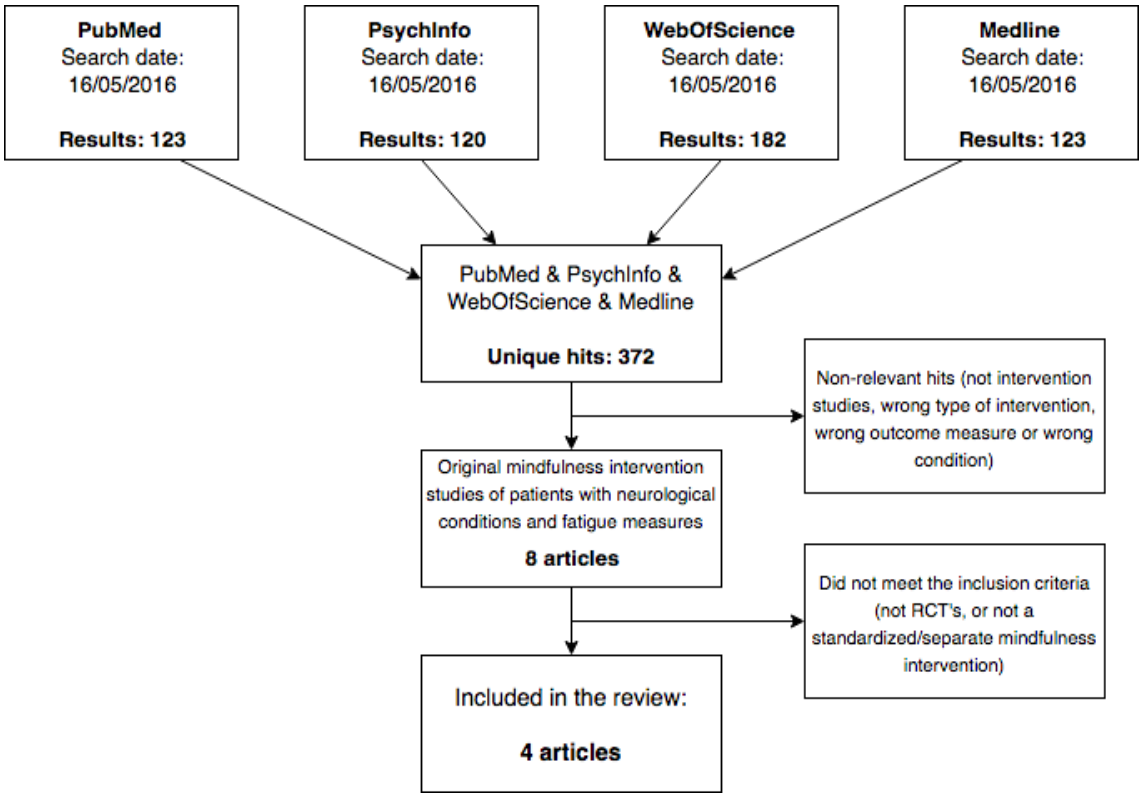

Figure 3.5 - Search process and study selection flowchart (adapted from [71])

The full dataset was requested to the authors, who returned a file containing references for 364 articles, instead of 372. From those 364 articles, and given that the article retrieval script (“PubMed Search and Save.py”) uses the PubMed search engine only, all the non-PubMed retrieved articles were taken out from the corpus. Four duplicated articles were also taken from the article list. This left the final corpus, i.e., the golden standard, with 119 articles divided into two classes: the four included in the SR, and the remaining 115 PubMed articles.

3.4.2. Humanin

Following the request of a researcher for help in their study regarding the Humanin protein, a dataset related to this problem was created from scratch.

To create the golden standard, a set of articles were retrieved from PubMed using the “humanin” keyword and a minimum publication date of 01/01/2012, using the PubMed Search and Save script explained in 3.3.1.

The resulting 85 articles were sent to the researcher for manual classification, i.e., the researcher was asked to classify from 1 to 5 how relevant each article was, by reading its title and abstract. With a numerical classification of 4 or 5, 60 articles were classified as “relevant,” and the remaining 25 articles (with numerical classifications from 1 to 3) as “non-relevant.” These were considered as the golden standard for classification purposes.

The full article list is available in table 1, in the annex E.

3.4. Practical Applications

This subsection presents two different practical applications for the classification script.

3.5.1. Mindfulness/Fatigue Dataset

The intent with Ulrichsen et al.’s [71] dataset was not to find articles belonging to one category or another, but instead finding a smaller amount of articles inside a bigger category, i.e., the SR-included articles inside the universe of PubMed retrieved articles of non-SR-included articles.

For this purpose, and since the goal was to study whether MeSH terms can be helpful in IR and article classification, the gold standard of 119 articles previously described in 3.4.1 was established as the test set. The reason for this is that since the labels for the 119 articles were known, a classification prediction could be validated as correct or incorrect. Two labels were created for the dataset: “SRincluded” for the SR-included articles, and “mindf_fatigue” for the remaining articles.

Several training sets were created following the guidelines from Ulrichsen et al.’s [71] SR, i.e.:

1. The inclusion criteria into the review seek to include “randomized (…) controlled trials aiming to measure the effect of different interventions on fatigue associated with neurological conditions and acquired brain injuries” and studies “primarily targeting fatigue” or including “fatigue as a secondary outcome measure.”
2. The exclusion criteria eliminated studies concerning fatigue “as a potential side effect of treatment, or as a contraindication for treatment (…), or studies targeting parallel, but different conditions to fatigue, such as sleepiness, reduced vigilance, anxiety, and

depression".

For the "SRincluded" articles, the exclusion criteria were approached using the "NOT" logical operator to exclude the undesired keywords. The usage of the "AND" logical operator ensured the retrieval of articles with both the desired keyword and a second set of keywords. The "OR" logical operator allows the user to retrieve several sets of keywords at once.

The queries given to the PubMed Search and Save script for the retrieval of the training sets were as presented by table 3.1. A maximum number of 500 articles per category was set for retrieval. No minimum date was set (i.e., "0000/01/01").

Table 3.1 - Queries for the mindfulness training set article retrieval

Trial	Query	Label
"Train"	(((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])) NOT ((adverse effects[MeSH Terms]) OR contraindications[MeSH Terms])	"SRincluded"
	(fatigue[MeSH Terms] AND mindfulness[MeSH Terms]) OR fatigue[MeSH Terms] OR mindfulness[MeSH Terms]	"mindf_fatigue"
"Train 1"	((((fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])) AND ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])) NOT ((((adverse effects[MeSH Terms]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms])	"SRincluded"
	(fatigue[MeSH Terms] AND mindfulness[MeSH Terms]) OR fatigue[MeSH Terms] OR mindfulness[MeSH Terms]	"mindf_fatigue"
"Train 2"	((((fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])) AND ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])) NOT ((((adverse effects[MeSH Terms]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms])	"SRincluded"
	(((((((fatigue[MeSH Terms]) AND mindfulness[MeSH Terms])) OR fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])) AND ((adverse effects[MeSH Subheading]) AND fatigue[MeSH Terms])) NOT randomized controlled trial[MeSH Terms]	"mindf_fatigue"
"Train 3"	(((((((fatigue[MeSH Terms]) AND mindfulness[MeSH Terms])) OR ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])))) NOT ((((adverse effects[MeSH Subheading]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms])))) OR ((((((fatigue[MeSH Terms]) AND mindfulness[MeSH Terms])) AND ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])))) NOT ((((adverse effects[MeSH Subheading]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms]))	"SRincluded"
	(((((((fatigue[MeSH Terms]) AND mindfulness[MeSH Terms])) OR fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])) AND ((adverse effects[MeSH Subheading]) AND fatigue[MeSH Terms])) NOT randomized controlled trial[MeSH Terms]	"mindf_fatigue"

For each corpus, that is, a training set (therefrom referred to according to their "trial" in table 3.1), and the golden standard as test set, several classification runs were performed with resource to the Classifier script, i.e., one for each classification algorithm.

3.5.2. Humanin Dataset

The problem which led to the creation of the Humanin dataset can be seen as a binary classification problem, that is, the categories to which a given article may belong are mutually exclusive.

For this purpose, and following the objective to study whether MeSH terms can be helpful in IR and article classification, the gold standard of 95 articles referred in 3.4.2 was established as the test set. Again, the reason for this is that since the labels for the 95 articles were known, a classification prediction could be validated as correct or incorrect. Two labels were created for the dataset: “relevant” for the Humanin-related articles, and “nonrelevant” for the remaining articles.

Unlike the mindfulness dataset, there were no previously dictated guidelines for the construction of the PubMed retrieval queries. The query-building strategy consisted of the combination of two approaches:

1. Starting with a set of keywords provided by the researcher (“activation”, “binding”, “Abeta”, “humanin”, “importin”, “brain”, “IGFBP3”, “TRIM11”, “BAX”, “BAK”, “bile acid”, “SHLP”, “MOTS-c”, “isoform”, “oligomerization”, “retrograde”, “humanin receptor”, “mitochondria”, “clinical”, “mutation”, “ubiquitin”, “microRNA”, “mtDNA”, “anaerobic”, “microbiome”, “apoptosis”, “cell survival”, “Alzheimer disease”), a small MeSH search was made to see which terms were available as descriptors.

   From the initial list, the terms available as descriptors in MeSH were the following: “humanin”, “mitochondria”, “mRNA”, “mtDNA”, “metabolism”, “peptides”, “apoptosis”, “cell survival”, “Alzheimer disease”, “brain”, “importin”, “mutation”, “ubiquitin”. These were saved as reference for future queries.

2. Starting with the main research term, “humanin,” a MeSH search was made with the intent of searching its hierarchy, i.e., the “parent” nodes, and using them as queries to generate training sets and evaluate the consequent performance.

   The referred MeSH search returned the tree represented in figure 3.6. The lettering in blue below each descriptor represents the number that indicates its tree location. Humanin itself does not have a tree location, as it is a supplementary concept rather than a descriptor.

Combining the knowledge gained from these two strategies, a set of queries was drawn. For the “non-relevant” to humanin articles, all humanin-related articles were excluded from the search using the “NOT humanin” logical operator. The inverse was made regarding the humanin “relevant” articles, i.e., the usage of the “AND humanin” logical operator to ensure the retrieval of articles with both the desired keyword and the humanin keyword itself. The “OR” logical operator

allows the user to retrieve several sets of keywords at once.

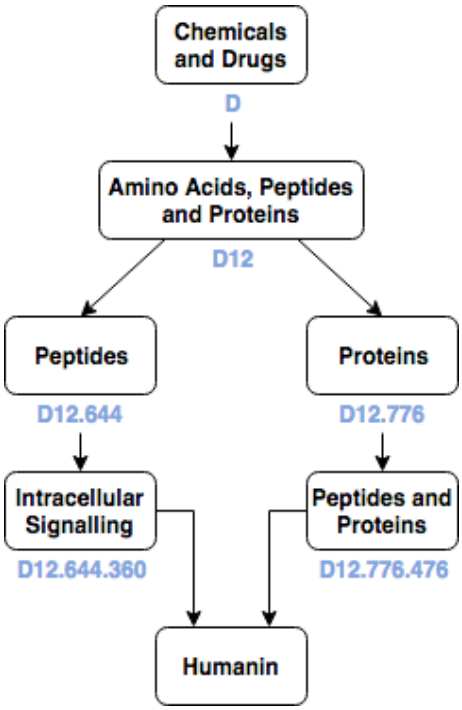

Figure 3.6 - Humanin MeSH hierarchy tree

As such, the queries given to the PubMed Search and Save script for the retrieval of the training sets were as presented by below according to table 3.2. A maximum number of 100 articles per category was set for retrieval. A minimum date of “2015/01/01” was set.

Table 3.2 - Queries for the Humanin training set article retrieval

Trial	Query	Label
“Train 1”	mitochondria[MeSH Terms] AND mtdna[MeSH Terms] AND peptides[MeSH Terms]	“nonrelevant”
	mitochondria[MeSH Terms] AND humanin	“relevant”
“Train 2”	mitochondria[MeSH Terms] AND mtdna[MeSH Terms] AND peptides[MeSH Terms]	“nonrelevant”
	(humanin) AND alzheimer disease[MeSH Terms]	“relevant”
“Train 3”	(importin[MeSH Terms] NOT humanin) OR (peptides[MeSH Terms] NOT humanin) OR (alzheimer's disease[MeSH Terms] NOT humanin) OR (brain[MeSH Terms] NOT humanin) OR (mutation[MeSH Terms] NOT humanin) OR (microrna[MeSH Terms] NOT humanin) OR (aging[MeSH Terms] NOT humanin) OR (cell survival[MeSH Terms] NOT humanin) OR (apoptosis[MeSH Terms] NOT humanin)	“nonrelevant”
	(humanin AND apoptosis[MeSH Terms) OR (humanin AND peptides[MeSH Terms]) OR (humanin AND cell survival[MeSH Terms]) OR (humanin AND aging[MeSH Terms]) OR (humanin AND microrna[MeSH Terms]) OR (humanin AND mutation[MeSH Terms]) OR (humanin AND brain[MeSH Terms]) OR (humanin AND alzheimer's disease[MeSH Terms]) OR (humanin AND mitochondria[MeSH Terms]) OR (humanin AND importin[MeSH Terms])	“relevant”
“Train 4”	((chemicals and drugs category[MeSH Terms])) NOT humanin	“nonrelevant”
	(humanin AND apoptosis[MeSH Terms) OR (humanin AND peptides[MeSH Terms]) OR (humanin AND cell survival[MeSH Terms]) OR (humanin AND aging[MeSH Terms]) OR (humanin AND microrna[MeSH Terms]) OR (humanin AND mutation[MeSH Terms]) OR	“relevant”

	(humanin AND brain[MeSH Terms]) OR (humanin AND alzheimer's disease[MeSH Terms]) OR (humanin AND mitochondria[MeSH Terms]) OR (humanin AND importin[MeSH Terms])	
"Train 5"	((chemicals and drugs category[MeSH Terms])) NOT humanin	"nonrelevant"
	((chemicals and drugs category[MeSH Terms])) AND humanin	"relevant"
"Train 6"	(amino acids, peptides, and proteins[MeSH Terms]) NOT humanin	"nonrelevant"
	(amino acids, peptides, and proteins[MeSH Terms]) AND humanin	"relevant"
"Train 7"	((peptides[MeSH Terms] AND proteins[MeSH Terms])) NOT humanin	"nonrelevant"
	((peptides[MeSH Terms] AND proteins[MeSH Terms])) AND humanin	"relevant"
"Train 8"	((intracellular signaling peptides and proteins[MeSH Terms])) NOT humanin	"nonrelevant"
	((intracellular signaling peptides and proteins[MeSH Terms])) AND humanin	"relevant"
"Train 9"	((peptides[MeSH Terms] AND proteins[MeSH Terms])) NOT humanin	"nonrelevant"
	((intracellular signaling peptides and proteins[MeSH Terms])) AND humanin	"relevant"
"Train 10"	(((anatomy category[MeSH Terms]) AND organisms category[MeSH Terms]) AND diseases category[MeSH Terms]) AND ((chemicals and drugs category[MeSH Terms]) NOT humanin) ((intracellular signaling peptides and proteins[MeSH Terms])) AND humanin	"nonrelevant"
	((intracellular signaling peptides and proteins[MeSH Terms])) AND humanin	"relevant"
"Train 11"	((anatomy category[MeSH Terms] OR organisms category[MeSH Terms] OR diseases category[MeSH Terms] OR chemicals and drugs category[MeSH Terms]) NOT humanin)	"nonrelevant"
	((intracellular signaling peptides and proteins[MeSH Terms])) AND humanin	"relevant"

For each corpus, that is, a training set (therefrom referred to according to their "trial" in table 3.2), and the golden standard as test set, several classification runs were performed with resource to the Classifier script, i.e., one for each classification algorithm.

Section 4

Results & Discussion

This section covers the results of the classification tasks, the model evaluations and the discussion of the different experiments.

4.1. Results

4.1.1. Mindfulness/Fatigue Dataset

Due to its extent, the classification reports for each corpus and classification algorithm can be consulted in Table 2, in the annex F. The average score of all classification trials ran with this dataset, that is, the average of both labels in each run, can be seen on table 4.1.

Table 4.1 - Average score of all classification trials with the mindfulness dataset

Algorithm	Average		
	Precision	Recall	F1-Score
Multinomial NB	66%	64%	56%
K Neighbors	67%	65%	55%
Random Forest	66%	63%	58%
Decision Trees	67%	69%	62%
Logistic Regression	67%	68%	61%

The classification algorithms that consistently achieved the best performance amongst all trials were Decision Trees (with an F1-score[26] of 62%) and Logistic Regression (with and F1-score of 61%). Nonetheless, when observing the individual label score values in table 2 (in the annex F), it can be noted that the “SRincluded” articles consistently achieve low scores.

A few examples are presented below, given their representation of the results of the classification trials. The first example, taking the Random Forest classification algorithm as reference, and trials “train” and “train 1”, is presented in table 4.2.

Table 4.2 - Classification reports for "train" and "train 1", using the Random Forest algorithm

Trial	Label	Precision	Recall	F1-Score	Conf. Matrix
Train	SRincluded	0%	0%	0%	[0 4] [9 106]
	mindf_fatigue	96%	92%	94%	
	avg / total	93%	89%	91%	

[26] As referred in 2.2.1.2, the harmonic mean of precision and recall, hence its usage as indicator of better/worse performance.

Train 1	SRincluded	5%	50%	9%	[2 2] [40 75]
	mindf_fatigue	97%	65%	78%	
	avg / total	94%	65%	76%	

The two training sets in question were chosen for comparison since their “mindf_fatigue” MeSH query is the same, i.e., the difference in both datasets resides in the “SRincluded” articles.

In the “train” trial, class “SRincluded” has no correctly classified articles; nonetheless, nine articles from the “mindf_fatigue” class were misclassified as “SRincluded.” “Mindf_fatigue” class achieved, as such, very high-performance scores in all analysed parameters. This is reflected in the average scores, as the number of articles corresponding to the “mindf_fatigue” is multiple times bigger than the number of “SRincluded” articles.

As for the “train 1” trial, it is seen that for the “SRincluded” articles there are now two correctly classified articles, that is, a recall of 50%. However, as 40 “mindf_fatigue” articles were misclassified as “SRincluded,” this resulted in a “SRincluded” precision score of only 5%. As for the “mindf_fatigue,” the main difference between the two trials resides in the recall and F-score scores, lowered by the misclassification of the 40 articles into the “SRincluded” class.

Again, as the number of articles corresponding to the “mindf_fatigue” is multiple times bigger than the number of “SRincluded” articles, the average scores reflect mostly the good performance of the first label.

Table 4.3 - Classification report for "train 3", using the Random Forest algorithm

Label	Precision	Recall	F1-Score	Conf. Matrix
SRincluded	13%	75%	22%	[3 1] [20 95]
mindf_fatigue	99%	83%	90%	
avg / total	96%	82%	88%	

Taking “train 3” individually as another example (chosen since it achieved the better classification for the “SRincluded” articles, across all other trials) classified using the Random Forest algorithm, it is seen that this time three out of four articles from the “SRincluded” class are correctly classified, with 20 articles from the “mindf_fatigue” class were misclassified as “SRincluded” as well - hence the high recall (75%) and low precision (13%) for this class.

4.1.1.1. Model Evaluation

For the evaluation of this model, the “train” training set is considered, together with the logistic regression classification algorithm (as presented in the first example in 4.1.1). All plots were generated by the Classifier script, together with the dataset classification.

The first evaluation measure to consider is the learning curve, presented below in figure 4.1. It can be seen that both the training and cross-validation scores increase, as the number of training examples increase as well, i.e., both scores are approximately 0.75 with a number of training scores below 100, and converge to a score value of approximately 0.9 when more than 600 training examples are available.

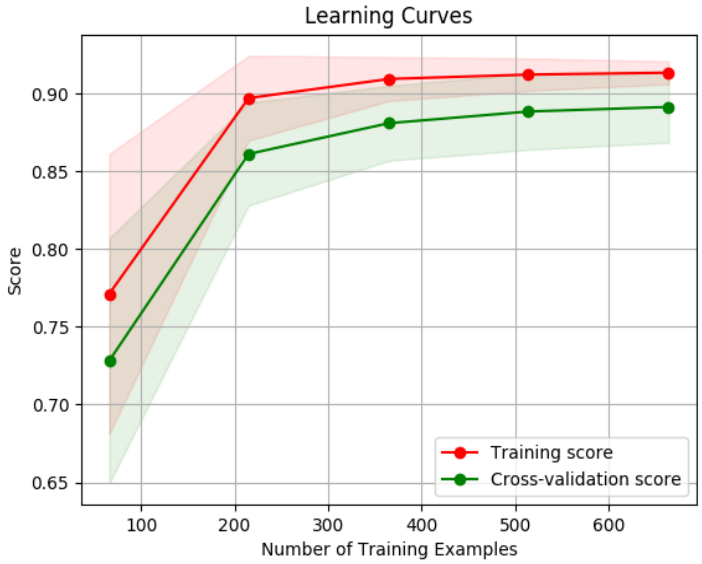

Figure 4.1 - Learning curves for the "train" training set and logistic regression classification algorithm

The next evaluation measure to consider is the ROC curve, to see how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples, and is presented below in figure 4.2.

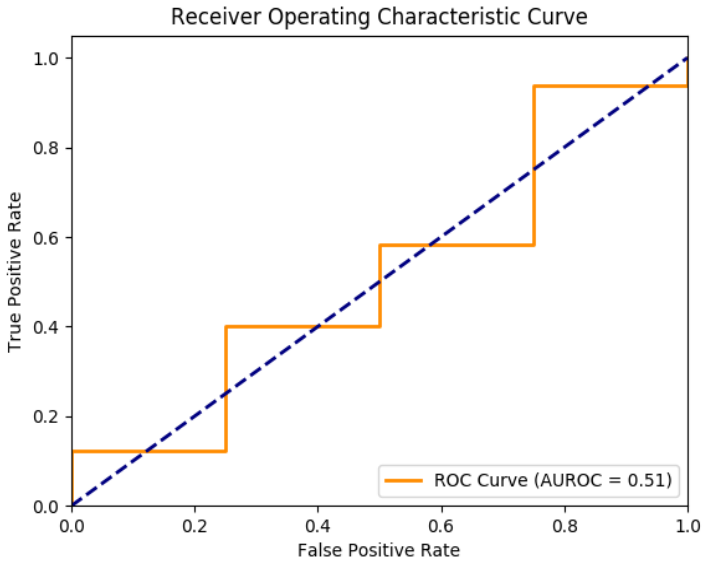

Figure 4.2 - ROC curve for the "train" training set and logistic regression classification algorithm

The generated plot shows a ROC curve beginning in a True Positive rate (i.e., the recall) slightly

below 0.2 and a False Positive rate of 0, and growing in “ladder”-type of increase, i.e., the growth of correctly classified examples is proportional to the growth of incorrectly classified negative examples.

The AUROC is 0.51, showing the probability of the classifier to assign a higher score to a randomly chosen positive example, rather than to a randomly chosen negative example.

The last model evaluation plot shows the precision-recall curve and is presented in figure 4.3.

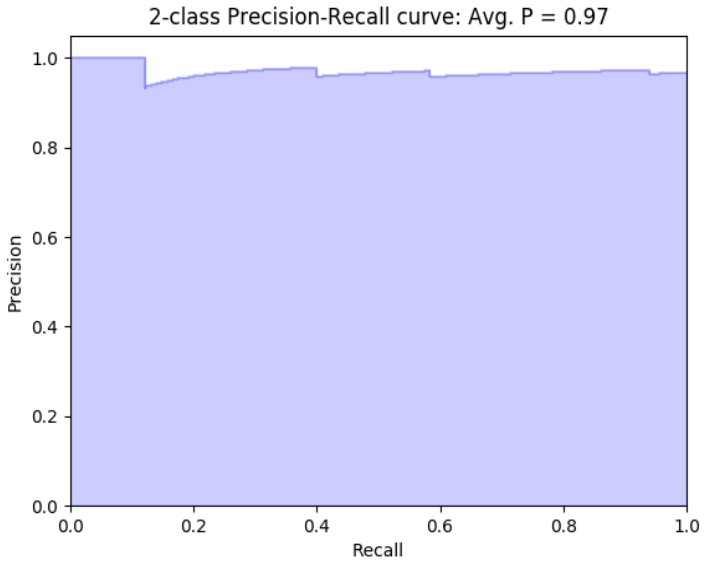

Figure 4.3 – Precision-recall curve for the "train" training set and logistic regression classification algorithm

The precision-recall curve shows the trade-off between precision and recall for different thresholds. In this case, the average precision is 0.97. It can be seen that there is a high area under the curve, denoting both high recall and high precision, i.e., a low false positive rate, as well as a low false negative rate. High scores for both measures indicate that the classifier is retrieving accurate results (i.e., high precision), as well as a majority of all positive results (i.e., high recall).

The results achieved in the PR curve presented above may be explained by the highly imbalanced dataset, on which one of the two classes being classified as consistently a high precision.

4.1.2. Humanin Dataset

Due to its extent, the classification reports for each corpus and classification algorithm can be consulted in Table 3, in the annex G. The average score of all classification trials ran with this dataset, that is, the average of both labels in each run, can be seen on table 4.4.

Table 4.4 - Average score of all classification trials with the humanin dataset

Algorithm	Average		
	Precision	Recall	F1-Score
Multinomial NB	70%	68%	68%
K Neighbors	70%	68%	68%
Random Forest	66%	62%	59%
Decision Trees	66%	62%	59%
Logistic Regression	76%	72%	73%

The classification algorithm that consistently achieved the best performance amongst all trials was Logistic Regression, with and F1-score of 73%. Again, as with the mindfulness dataset case, when observing the individual label score values in table 3 (in the annex G), it can be seen that there is a gap in the performance scores of both labels – the “nonrelevant” class consistently achieves low performance scores, and the “relevant” class consistently achieves good scores.

A few examples are presented below, given their representation of the overall results of the classification trials. The first example, taking the Logistic Regression classification algorithm as reference, and trials “train 4” and “train 5”, is presented in table 4.5.

Table 4.5 - Classification reports for "train 4" and "train 5”, using the Logistic Regression algorithm

Trial	Label	Precision	Recall	F1-Score	Conf. Matrix
Train 4	nonrelevant	80%	48%	60%	[12 13] [3 57]
	relevant	81%	95%	88%	
	avg / total	81%	81%	80%	
Train 5	nonrelevant	77%	40%	53%	[10 15] [3 57]
	relevant	79%	95%	86%	
	avg / total	79%	79%	76%	

The two training sets in question were chosen for comparison since their “nonrelevant” MeSH query is the same, i.e., the difference in both datasets resides in the “relevant” articles.

In the “train 4” trial, and looking at the confusion matrix, it is seen that approximately half of the “nonrelevant” articles (12 out of 25) were correctly classified, hence the recall value of 48%. Nonetheless, since three articles from the “relevant” class were misclassified as “nonrelevant,” the precision of the latter class achieves a value of 80%. As for the “relevant” labelled articles, 57 articles were correctly classified, and 13 were misclassified as “nonrelevant,” hence the 81% precision value. The 95% recall score is due to three out of 60 articles for this class being incorrectly classified.

As the number of articles corresponding to the “nonrelevant” class is significantly different from the number of “relevant” class articles, i.e., 25 versus 60 articles, the average scores reflect mostly the performance of the “relevant” class, hence the average 80% F1-score.

Regarding the “train 5” trial, the difference in the performance scores is given to the lowering of the precision score for the “nonrelevant” class, from 80% in “trial 4” to 77% in “trial 5”, that is, a difference of two incorrectly classified articles. This difference slightly affects all the remaining performance scores negatively.

The second example, presented in the table 4.6, is representative of the majority of the trials ran with other training sets and classification algorithms.

Table 4.6 - Classification report for "train 2", using the K-Neighbors algorithm

Label	Precision	Recall	F1-Score	Conf. Matrix
nonrelevant	55%	64%	59%	[16 9] [13 47]
relevant	84%	78%	81%	
avg / total	75%	74%	75%	

Looking at the confusion matrix, it can be seen that there is a slightly greater number of “nonrelevant” articles correctly classified (16 out of 25) when compared to the examples presented by table 4.5. At the same time, there is also a greater number of misclassifications in the “relevant” class articles, that is, 13 from the 60 articles belonging to this class were incorrectly classified as “nonrelevant.” Thus, the resulting scores: a precision of 55% for the “nonrelevant” class, given its correct classification of 16 articles as “nonrelevant” and incorrect classification of another 13 articles as “nonrelevant”, and a 64% recall score for the same label, given the 16 correctly classified articles classified as “nonrelevant” and 9 misclassified as “relevant”.

As for the “relevant” label, with 47 out of 60 articles correctly classified, the precision score achieved a value of 84%, and the recall score of 78% is explained by the nine articles from the “nonrelevant” class incorrectly classified as “relevant.”

Again, given the difference of number of articles from both labels, the average scores reflect mostly the performance of the “relevant” class, hence the average 75% F1-score.

4.1.2.1. Model Evaluation

For the evaluation of this model, the “train 4” training set is considered, together with the logistic regression classification algorithm (as presented in the table 4.5). All plots are generated by the Classifier script, together with the dataset classification.

The first evaluation measure to consider is the learning curve, presented below in figure 4.4. It

can be seen that both training and cross-validation scores increase, as the number of training examples increase as well, i.e., CV score starts at a 0.7 score with 50 training examples and training score is slightly above 0.75 for the same number of training, and both measures converge to a score value of approximately 0.95 when more than 400 training examples are available.

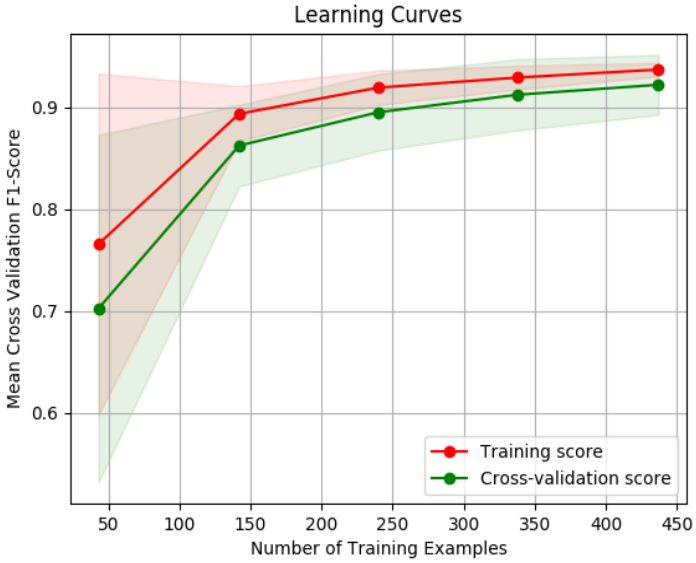

Figure 4.4 - Learning curves for the "train 4" training set and logistic regression classification algorithm

The next evaluation measure to consider is the ROC curve, to see how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples, and is presented below in figure 4.5.

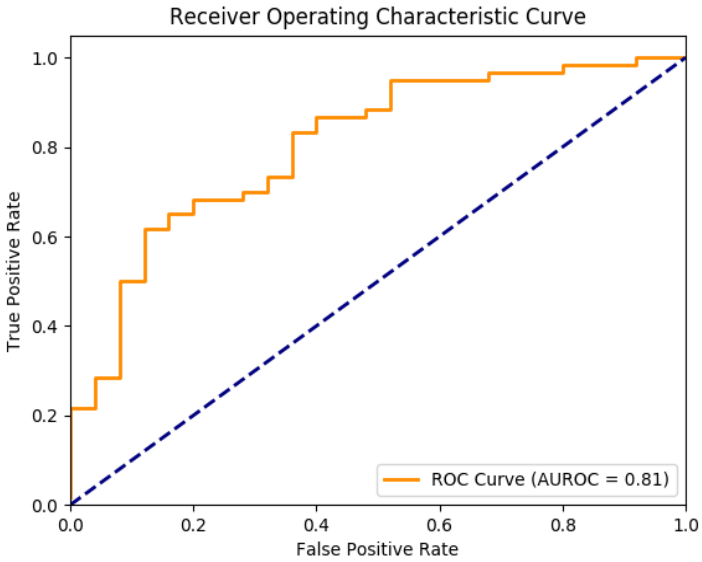

Figure 4.5 - ROC curve for the "train 4" training set and logistic regression classification algorithm

The generated plot shows a ROC curve beginning in a True Positive rate (i.e., the recall) slightly below 0.2 and a False Positive rate of 0, growing consistently in the upper-left side of the plot until

it starts to stagnate at a False Positive rate of 0.5, near the True Positive rate value of 1. The AUROC is 0.81, showing the probability of the classifier to assign a higher score to a randomly chosen positive example, rather than to a randomly chosen negative example.

The last model evaluation plot shows the precision-recall curve and is presented in figure 4.6.

The precision-recall curve shows the trade-off between precision and recall for different thresholds. In this case, the average precision is 0.91. It can be seen that there is a high area under the curve, even though the curve as a negative slope, i.e., overall, the precision value decreases as the recall value increases. This indicates that as the number of positive results (i.e., the recall) increases, the classifier may retrieve less accurate results.

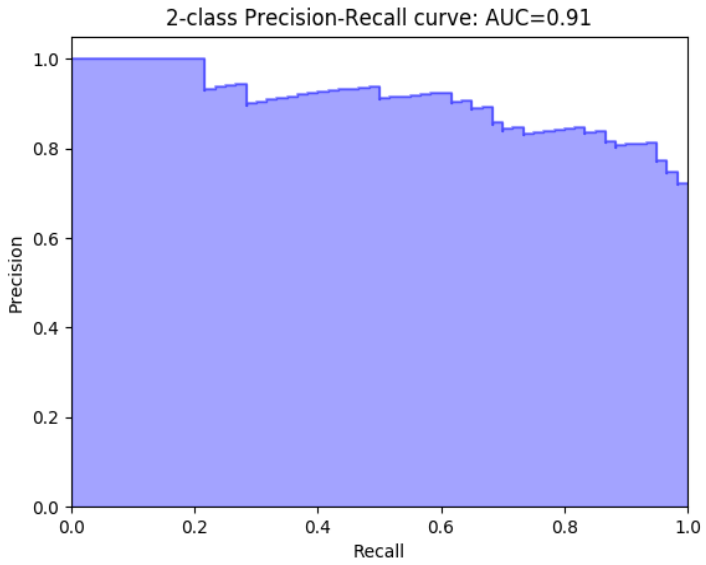

Figure 4.6 - Precision/recall curve for the "train 4" training set and logistic regression classification algorithm

4.2. Discussion

The analysed datasets presented different challenges for the proposed task, which may help explain the obtained results.

The mindfulness/fatigue dataset, as explained in 3.4.1, was based in the SR by Ulrichsen et al. [71]. Given that one of the classes enclosed only four articles (the ones included in the SR), the final dataset turned out to be highly imbalanced.

As such, the majority of the classification trials achieve low-performance scores for the “SRincluded” class, and when it does not, this is achieved at the expense of a greater number of misclassifications in the other class articles.

This can be verified by the results shown by table 4.2. As referred in 4.1.1, the two training sets

were chosen for comparison since the MeSH terms chosen for the “mindf_fatigue” PubMed query is the same (“(fatigue[MeSH Terms] AND mindfulness[MeSH Terms]) OR fatigue[MeSH Terms] OR mindfulness[MeSH Terms]”, i.e., the difference in both datasets resides in the “SRincluded” articles. The table 4.7 presents the different queries for both trials.

Table 4.7 - Queries for the "SRincluded" class from the Mindfulness/fatigue dataset

Trial	Query
Train	(((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])) NOT ((adverse effects[MeSH Terms]) OR contraindications[MeSH Terms])
Train 1	((((fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])) AND ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])) NOT ((((adverse effects[MeSH Terms]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms])

As seen, “train 1” query shows an extension of the “train” query, as it adds specificity for the articles retrieved through the addition of the “(fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])” and “OR anxiety[MeSH Terms]) OR depression[MeSH Terms]” to the query. Raising the specificity of the “SRincluded” label resulted in the correct classification of two out of the four articles for that class, but at the same time, it also raised the recall score from 0 to 50%.

However, “train 3”, as presented by table 4.3, achieved much better performance scores. The reason is behind the MeSH terms chosen for the PubMed queries, which can be seen in table 4.8.

Table 4.8 - MeSH terms for the "train 3" PubMed queries

Class	Query
“SRincluded”	(((((((fatigue[MeSH Terms]) AND mindfulness[MeSH Terms])) OR ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms]))) NOT ((((adverse effects[MeSH Subheading]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms]))) OR ((((((fatigue[MeSH Terms])

	AND mindfulness[MeSH Terms])) AND ((randomized controlled trial[MeSH Terms]) OR brain injuries[MeSH Terms])))) NOT ((((adverse effects[MeSH Subheading]) OR contraindications[MeSH Terms]) OR anxiety[MeSH Terms]) OR depression[MeSH Terms]))
“Mindf_fatigue”	(((((((fatigue[MeSH Terms]) AND mindfulness[MeSH Terms])) OR fatigue[MeSH Terms]) OR mindfulness[MeSH Terms])) AND ((adverse effects[MeSH Subheading]) AND fatigue[MeSH Terms])) NOT randomized controlled trial[MeSH Terms]

As it can be seen, the usage of more MeSH terms, aided by the usage of the “AND/OR/NOT” logical operators, to create more complex and comprehensive queries may help improve the performance scores of this kind of classification problems. This statement is corroborated by the results achieved by the humanin dataset.

Looking at the “train 4” e “train 5”, presented by table 4.5, it can be seen that the “train 4” trial achieved a better classification performance. When looking at the MeSH terms used to build the PubMed queries, it can be seen why: the “relevant” class for the “train 4” had a much more complex query than the same class for the “train 5”, as presented by table 4.9.

Table 4.9 - Queries for the "relevant" class from the Humanin dataset

Trial	Query
Train 4	(humanin AND apoptosis[MeSH Terms) OR (humanin AND peptides[MeSH Terms]) OR (humanin AND cell survival[MeSH Terms]) OR (humanin AND aging[MeSH Terms]) OR (humanin AND microrna[MeSH Terms]) OR (humanin AND mutation[MeSH Terms]) OR (humanin AND brain[MeSH Terms]) OR (humanin AND alzheimer's disease[MeSH Terms]) OR (humanin AND mitochondria[MeSH Terms]) OR (humanin AND importin[MeSH Terms])
Train 5	((chemicals and drugs category[MeSH Terms])) AND humanin

It shall be remembered that, as referred in 3.5.2, one of the query-building strategies involved starting with the main research term, “humanin,” and then using the “parent” nodes to generate new queries in order to evaluate the consequent performance.

With the results obtained, and as referred before, one can state that the usage of more MeSH terms to create more complex and comprehensive queries may help improve the performance scores of this kind of classification problems.

Section 5
Conclusions & Future Work

This section summarises the conclusions of this work, discusses some limitations and future work ideas, and presents the final remarks.

5.1. Summary

The main objective of this work was to test the hypothesis that TM tools and controlled vocabularies have a positive impact on the systematic reviewing, either from an aspect of time reduction or regarding performance (i.e., if a given article is relevant to the study or not).

For the accomplishment of this objective, a system capable of creating a classification model which training is based on a controlled vocabulary (MeSH) that can be applied to a variety of biomedical literature was developed. The aim was not to (re-)create any existing algorithms, but to study whether this approach would have an impact on the SR process.

As stated by several authors ([9], [42], [53], [55], [58], among others), the strength of automatizing the systematic reviewing process resides mostly in a significant reduction of time spent describing studies (versus a manual verification), but also enabling studies to be described according to an external framework.

The usage of two different datasets, with two completely different origins, allowed to see the behavior of this approach in two different scenarios and evaluate the usage of TM tools aligned with controlled vocabularies (as is MeSH).

Though the findings may have been limited by the datasets used, the limitations found may as well be the starting point for further studies.

5.1.1. Limitations

The task on which this work focuses brings up several challenges. One of the most challenging limitations found is the application of exclusion/inclusion (eligibility) criteria, which was the case with the mindfulness dataset results. A suggestion for further improvement may reside either in the refinement of stopwords or, in a long stretch, the creation of a controlled vocabulary for the PICOS (Participants, Interventions, Comparators, Outcomes and Study design) inclusion criteria, as this is the most applied methodology.

As another limitation example, consider a search for a conceptually broad review, necessarily wide in scope. The usage of the MeSH terms (or any controlled vocabulary) for the creation of the

search query and consequent training data will probably have no influence on the result, as the system will retrieve a large number of irrelevant articles, in comparison to the number of correct hits.

There is also the detail that the chosen vocabulary may not include the concepts needed in order to retrieve all the desired articles. On the other hand, using controlled vocabularies to narrow down the retrieved articles may be a powerful way of finding documents quickly, but it is possible that the method will miss potentially relevant studies that, if manually observed, wouldn't be missed.

Another flaw is that when using third-party software, as is Scikit-learn, the user is limited to what it offers. In this case, the confusion matrices used to show how many articles are correctly/incorrectly classified are created in a way that the matrix does not store each article's name. This results in the loss of some information about each article and its classification, retrieving only the number of correctly/incorrectly classified articles.

5.1.2. Final Remarks

Systematic reviews are considered today a widely accepted research method. However, as medical knowledge increases (and, with it, the amount of literature published every day), it is increasingly difficult to conduct them to fit with policy and practice timescales. This is especially true in areas of study which databases are non-comprehensive and inconsistently-indexed.

Given that the ultimate desire in the area is that studies may be included or excluded without the need to ever being seen by a human, a few questions arise: should TM tools reach perfect precision scores so that they can be used in systematic reviews? Moreover, even if a given tool has shown to be 100% precise in for a given study/SR, how can the final user be sure that the score achieved in that review will apply to any other study?

One may argue that TM tools, which are ever-changing and improving, may not need to claim perfect performance scores if they can demonstrate success in solving some of the problems reviewers currently face. More specifically, if this results in reducing the length of time that it takes to identify the studies that will ultimately be included in the review. However, this may raise a conceptual challenge to reviewers: assuming that there is a big difference between not having retrieved a study (as it is nearly impossible to search everything) and having retrieved it but excluding it inaccurately as the result of an automatic process, will stakeholders accept a method that clearly declares that, for example, 5% of studies retrieved are incorrectly excluded?

A good strategy for reviewers may be the application of a 'multi-layered' way of finding relevant research. That is, if one considers with each layer (database searching, hand searching, looking for citations, contacting authors, among other tasks) intended to make up for deficiencies in other

layers. Following this line of thinking, the benefits that both TM tools and controlled vocabularies offer may more than outweigh any associated or perceived deficiencies.

As TM tools and controlled vocabularies are now being employed in different areas, extra efforts towards methodological and evaluative work may be required to develop methods and an evidence base for their use. This work was developed hoping it would leave its contribution to this path.

With all this being said, and as a final note, it is believed that researchers and scientists would deeply benefit from training, both to manage expectations and to ensure that systematic reviewers understand the benefits but also the limitations of TM tools, whenever their scope is. The correct adoption of TM within SR is believed to depend greatly on cooperation between systematic reviewers and computer scientists, so that customised and optimised solutions may thrive.

Bibliography

[1] B. A. Barron and S. C. Bukantz, “The evaluation of new drugs: Current food and drug administration regulations and statistical aspects of clinical trials,” *Arch. Intern. Med.*, vol. 119, no. 6, pp. 547–556, Jun. 1967.

[2] B. L. Humphreys and D. E. McCutcheon, “Growth patterns in the National Library of Medicine’s serials collection and in Index Medicus journals, 1966-1985.,” *Bull. Med. Libr. Assoc.*, vol. 82, no. 1, pp. 18–24, Jan. 1994.

[3] H. Bastian, P. Glasziou, and I. Chalmers, “Seventy-five trials and eleven systematic reviews a day: How will we ever keep up?,” *PLoS Med.*, vol. 7, no. 9, 2010.

[4] S. Jaeger, S. Gaudan, U. Leser, and D. Rebholz-Schuhmann, “Integrating protein-protein interactions and text mining for protein function prediction,” *BMC Bioinformatics*, vol. 9, no. 8, p. S2, Jul. 2008.

[5] N. Papanikolaou, G. A. Pavlopoulos, T. Theodosiou, I. S. Vizirianakis, and I. Iliopoulos, “DrugQuest - a text mining workflow for drug association discovery,” *BMC Bioinformatics*, vol. 17, no. S5, p. 182, 2016.

[6] G. Tsafnat, P. Glasziou, M. K. Choong, A. Dunn, F. Galgani, and E. Coiera, “Systematic review automation technologies,” *Syst. Rev.*, vol. 3, no. 1, p. 74, 2014.

[7] J. Higgins and S. Green, “Cochrane Handbook for Systematic Reviews of Interventions,” in *The Cochrane Collaboration*, 2011.

[8] S. R. Jonnalagadda, P. Goyal, and M. D. Huffman, “Automating data extraction in systematic reviews: a systematic review,” *Syst. Rev.*, vol. 4, no. 1, p. 78, 2015.

[9] A. O’Mara-Eves, J. Thomas, J. McNaught, M. Miwa, and S. Ananiadou, “Using text mining for study identification in systematic reviews: a systematic review of current approaches,” *Syst. Rev.*, vol. 4, no. 1, p. 5, 2015.

[10] A.-H. Tan, “Text Mining: The state of the art and the challenges,” *Proc. PAKDD 1999 Work. Knowl. Discov. from Adv. Databases*, vol. 8, pp. 65–70, 1999.

[11] A. Hotho, A. Nürnberger, and G. Paaß, “A Brief Survey of Text Mining,” *LDV Forum - Gld. J. Comput. Linguist. Lang. Technol.*, vol. 20, pp. 19–62, 2005.

[12] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, vol. 35, no. 2. 2008.

[13] A. Lamurias, L. A. Clarke, and F. M. Couto, “Extracting microRNA-gene relations from biomedical literature using distant supervision,” *PLoS One*, vol. 12, no. 3, 2017.

[14] M. Krallinger, O. Rabal, A. Lourenço, J. Oyarzabal, and A. Valencia, “Information retrieval and text mining technologies for chemistry,” *Chem. Rev.*, vol. 117, no. 12, pp. 7673–7761, 2017.

[15] S. Ananiadou, “Text Mining for Biomedicine,” *Information retrieval in biomedicine: natural …*, vol. 10, no. 6. Artech House, Boston/London, pp. 1–11, 2009.

[16] J. J. Webster and C. Kit, “Tokenization as the initial phase in NLP,” in *Proceedings of the 14th conference on Computational linguistics* -, 1992, vol. 4, p. 1106.

[17] A. Leal, “Recognition and Normalization of Biomedical Entities Within Clinical Notes,” p.

123, 2015.

[18] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[19] H. Liu, T. Christiansen, W. A. Baumgartner, and K. Verspoor, “BioLemmatizer: a lemmatization tool for morphological processing of biomedical text,” *J. Biomed. Semantics*, vol. 3, p. 3, Apr. 2012.

[20] A. L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, 1959.

[21] M. Allahyari *et al.*, “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques,” 2017.

[22] W. Zhang and F. Gao, “An improvement to naive bayes for text classification,” *Procedia Eng.*, vol. 15, pp. 2160–2164, 2011.

[23] C. D. Manning, P. Ragahvan, and H. Schutze, “An Introduction to Information Retrieval,” *Inf. Retr. Boston.*, no. c, pp. 1–18, 2009.

[24] H. Zhang, “The Optimality of Naive Bayes,” *Proc. Seventeenth Int. Florida Artif. Intell. Res. Soc. Conf. FLAIRS 2004*, vol. 1, no. 2, pp. 1–6, 2004.

[25] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. 2010.

[26] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[27] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *Am. Stat.*, vol. 46, no. 3, pp. 175–185, 1992.

[28] Z. Zhang, “Introduction to machine learning: k-nearest neighbors,” *Ann. Transl. Med.*, vol. 4, no. 11, pp. 218–218, 2016.

[29] P. Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. 2012.

[30] L. Rokach and O. Maimon, *Data Mining With Decision Trees - Theory and Applications*, 2nd ed. World Scientific Publishing Co. Pte. Ltd., 2015.

[31] T. K. Ho, “Random decision forests,” *Proc. 3rd Int. Conf. Doc. Anal. Recognit.*, vol. 1, pp. 278–282, 1995.

[32] S. H. Walker and D. B. Duncan, “Estimation of the Probability of an Event as a Function of Several Independent Variables,” *Biometrika*, vol. 54, no. 1/2, p. 167, 1967.

[33] D. A. Freedman, “Statistical Models: Theory and Practice,” *Cambridge Univ. Press*, p. 442, 2009.

[34] S. Knerr, L. Personnaz, and G. Dreyfus, “Single-layer learning revisited: a stepwise procedure for building and training a neural network,” in *Neurocomputing*, 1990, pp. 41–50.

[35] A. M. Cohen and W. R. Hersh, “A survey of current work in biomedical text mining,” *Br. Bioinform*, vol. 6, no. 1, pp. 57–71, 2005.

[36] F. Provost, T. Fawcett, and R. Kohavi, “The Case Against Accuracy Estimation for Comparing Induction Algorithms,” *Proc. Fifteenth Int. Conf. Mach. Learn.*, pp. 445–453, 1997.

[37] Z. E. Rasjid and R. Setiawan, “Performance Comparison and Optimization of Text Document Classification using k-NN and Naïve Bayes Classification Techniques,” *Procedia Comput. Sci.*, vol. 116, pp. 107–112, 2017.

[38] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves,” *Proc. 23rd Int. Conf. Mach. Learn. - ICML ’06*, pp. 233–240, 2006.

[39] M. Stone, “Cross-Validatory Choice and Assessment of Statistical Predictions,” *J. R. Stat. Soc.*, vol. 36, no. 2, pp. 111–147, 1974.

[40] C. Determan Jr., “Cross validation with test data set,” *Cross Validated*, 2017. [Online]. Available: https://stats.stackexchange.com/a/148698. [Accessed: 17-Oct-2017].

[41] G. C. Cawley and N. L. C. Talbot, “On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation,” *J. Mach. Learn. Res.*, vol. 11, p. 2079−2107, 2010.

[42] J. Thomas, J. McNaught, and S. Ananiadou, “Applications of text mining within systematic reviews,” *Res. Synth. Methods*, vol. 2, no. 1, pp. 1–14, 2011.

[43] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

[44] N. Guarino, “Formal Ontology and Information Systems,” *Proc. first Int. Conf.*, no. June, pp. 3–15, 1998.

[45] D. Kless, “The differences and similarities in thesaurus and ontology structure: with a method for reengineering thesauri into qualitatively good ontologies,” 2014.

[46] National Information Standards Organization, “ANSI/NISO Z39.19-2005: Guidelines for the Construction , Format , and Management of Monolingual Controlled Vocabularies,” 2005.

[47] NLM, “MeSH Preface,” 2016. [Online]. Available: https://www.nlm.nih.gov/mesh/intro_preface.html#pref_hist. [Accessed: 11-Jun-2018].

[48] F. B. Rogers, “Medical subject headings.,” *Bull. Med. Libr. Assoc.*, vol. 51, no. January, pp. 114–116, 1963.

[49] I. Dhammi and S. Kumar, “Medical subject headings (MeSH) terms,” *Indian J. Orthop.*, vol. 48, no. 5, p. 443, 2014.

[50] NLM, “MeSH Record Types,” 2017. [Online]. Available: https://www.nlm.nih.gov/mesh/intro_record_types.html. [Accessed: 11-Jun-2018].

[51] NLM, “MeSH Tree Structures,” 2016. [Online]. Available: https://www.nlm.nih.gov/mesh/intro_trees.html. [Accessed: 12-Jun-2018].

[52] NLM, “Use of MeSH in Online Retrieval,” 2014. [Online]. Available: https://www.nlm.nih.gov/mesh/intro_retrieval.html. [Accessed: 12-Jun-2018].

[53] Y. Aphinyanaphongs, I. Tsamardinos, A. Statnikov, D. Hardin, and C. F. Aliferis, “Text categorization models for high-quality article retrieval in internal medicine,” *J. Am. Med. Informatics Assoc.*, vol. 12, no. 2, pp. 207–216, Mar. 2005.

[54] P. Pluye, R. M. Grad, L. G. Dunikowski, and R. Stephenson, “Impact of clinical information-retrieval technology on physicians: A literature review of quantitative, qualitative and mixed

methods studies," *Int. J. Med. Inform.*, vol. 74, no. 9, pp. 745–768, 2005.

[55] R. Paynter *et al.*, "EPC Methods: An Exploration of the Use of Text-Mining Software in Systematic Reviews," p. 70, 2016.

[56] B. K. Olorisade, P. Brereton, and P. Andras, "Reproducibility of studies on text mining for citation screening in systematic reviews: Evaluation and checklist," *J. Biomed. Inform.*, vol. 73, pp. 1–13, 2017.

[57] R. Paynter, L. L. Bañez, E. Erinoff, J. Lege-Matsuura, and S. Potter, "Commentary on EPC methods: an exploration of the use of text-mining software in systematic reviews," *J. Clin. Epidemiol.*, vol. 84, pp. 33–36, 2017.

[58] J. Rathbone, T. Hoffmann, and P. Glasziou, "Faster title and abstract screening? Evaluating Abstrackr, a semi-automated online screening program for systematic reviewers," *Syst. Rev.*, vol. 4, no. 1, p. 80, 2015.

[59] P. F. Balan, A. Gerits, and W. Vanduffel, "A practical application of text mining to literature on cognitive rehabilitation and enhancement through neurostimulation," *Front. Syst. Neurosci.*, 2014.

[60] R. Kok, J. A. H. M. Verbeek, B. Faber, F. J. H. Van Dijk, and J. L. Hoving, "A search strategy to identify studies on the prognosis of work disability: A diagnostic test framework," *BMJ Open*, 2015.

[61] E. Hausner, S. Waffenschmidt, T. Kaiser, and M. Simon, "Routine development of objectively derived search strategies," *Syst. Rev.*, 2012.

[62] M. Petrova, P. Sutcliffe, K. W. M. Fulford, and J. Dale, "Search terms and a validated brief search filter to retrieve publications on health-related values in Medline: A word frequency analysis study," *J. Am. Med. Informatics Assoc.*, 2012.

[63] A. O'Mara-Eves, G. Brunton, D. Mcdaid, J. Kavanagh, S. Oliver, and J. Thomas, "Techniques for identifying cross-disciplinary and 'hard-to-detect' evidence for systematic review," *Res. Synth. Methods*, 2014.

[64] P. J. A. Cock *et al.*, "Biopython: Freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[65] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, vol. 43. 2009.

[66] R. Bilbro, "An Introduction To Machine Learning With Python." [Online]. Available: https://www.districtdatalabs.com/an-introduction-to-machine-learning-with-python.

[67] Z. S. Harris, "Distributional Structure," WORD, vol. 10, no. 2–3, pp. 146–162, 1954.

[68] J. Shaikh, "Machine Learning, NLP: Text Classification using scikit-learn, python and NLTK," 2017. [Online]. Available: https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a.

[69] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, 2006.

[70] L. F. Campos, A. Lamurias, and F. M. Couto, "Can the Wisdom of the Crowd Be Used to Improve the Creation of Gold-standard for Text Mining applications?," *To Appear proceddings 9th INForum - Simpósio Informática (INForum 2017).*, 2017.

[71] K. M. Ulrichsen *et al.*, "Clinical utility of mindfulness training in the treatment of fatigue after

stroke, traumatic brain injury and multiple sclerosis: A systematic literature review and meta-analysis," *Front. Psychol.*, vol. 7, no. JUN, pp. 1–11, 2016.

[72] D. Parkes, "The ROC Curve," 2018. [Online]. Available: https://deparkes.co.uk/2018/02/16/the-roc-curve/. [Accessed: 29-Aug-2018].

Annex

A. ROC Curve Example

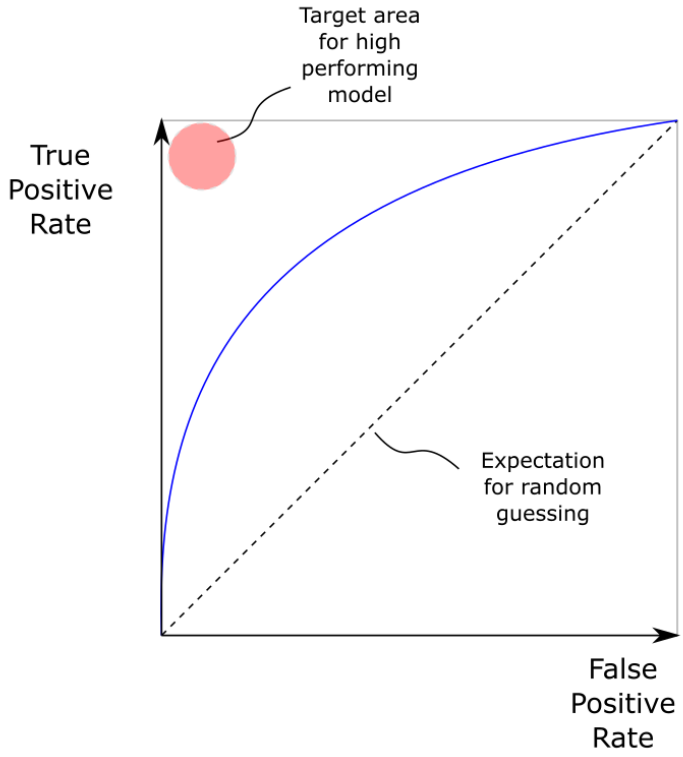

Figure 1 - Example and explanation of a ROC curve (adapted from [72])

B. MeSH Browser Search Example

Brain MeSH Descriptor Data 2017

Details | Qualifiers | MeSH Tree Structures | Concepts

MeSH Heading	Brain
Tree Number(s)	A08.186.211
Unique ID	D001921
Annotation	general; /blood supply: consider also CEREBROVASCULAR CIRCULATION; CEREBRAL ARTERIES; CEREBRAL VEINS; CRANIAL SINUSES; /cytol: do not routinely convert to NEURONS; /surg: consider specific neurosurgical procedures in Category E4; inflammation = ENCEPHALITIS & its specifics; infarct = CEREBRAL INFARCTION; malacia = ENCEPHALOMALACIA; brain-isolated, encéphale isolé, cerveau isolé: index DECEREBRATE STATE
Scope Note	The part of CENTRAL NERVOUS SYSTEM that is contained within the skull (CRANIUM). Arising from the NEURAL TUBE, the embryonic brain is comprised of three major parts including PROSENCEPHALON (the forebrain); MESENCEPHALON (the midbrain); and RHOMBENCEPHALON (the hindbrain). The developed brain consists of CEREBRUM; CEREBELLUM; and other structures in the BRAIN STEM.
Entry Term(s)	Encephalon
See Also	Cerebral Decortication Psychosurgery
Consider Also	consider also terms at CEREBR- and ENCEPHAL-
Public MeSH Note	/enzymology was BRAIN ENZYMOLOGY 1964-65; /physiology was BRAIN PHYSIOLOGY 1965; BRAIN ELECTROPHYSIOLOGY was heading 1964-65
Online Note	pre-explosion = BRAIN (PX)
History Note	/enzymology was BRAIN ENZYMOLOGY 1964-65; /physiology was BRAIN PHYSIOLOGY 1965; BRAIN ELECTROPHYSIOLOGY was heading 1964-65
Entry Combination	chemistry:Brain Chemistry injuries:Brain Injuries transplantation:Brain Tissue Transplantation
Date Established	1960/01/01
Date of Entry	1999/01/01
Revision Date	2016/08/09

Figure 2 - MeSH browser example, using the search term "brain"

C. Model Evaluation – Bag of Words

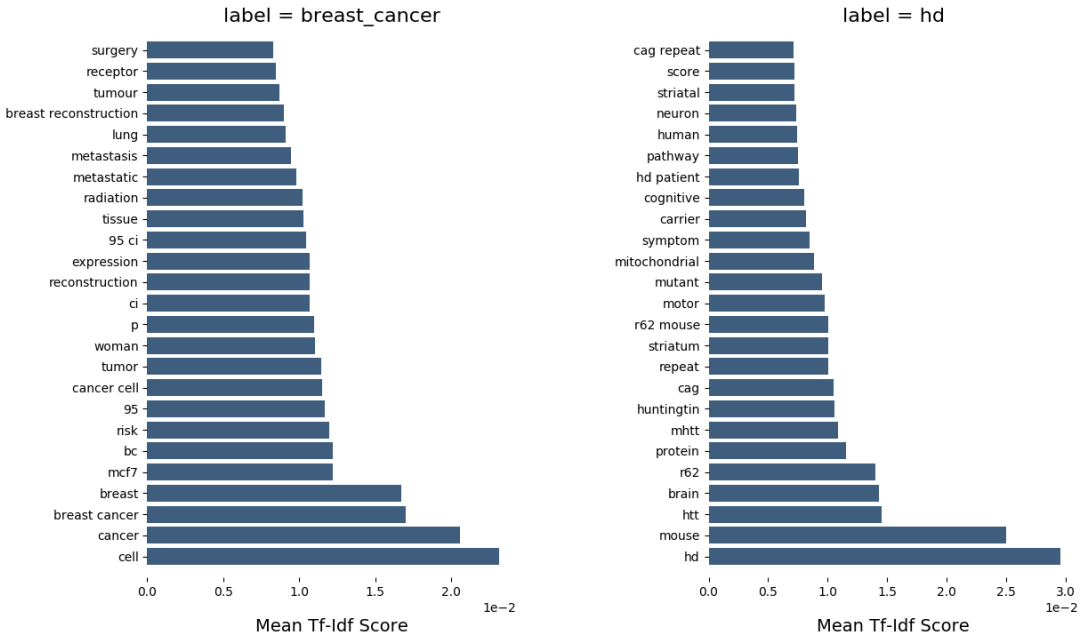

Figure 3 - Bag of words and Tf-Idf score for each word and label

D. Model Evaluation – Confusion Matrix

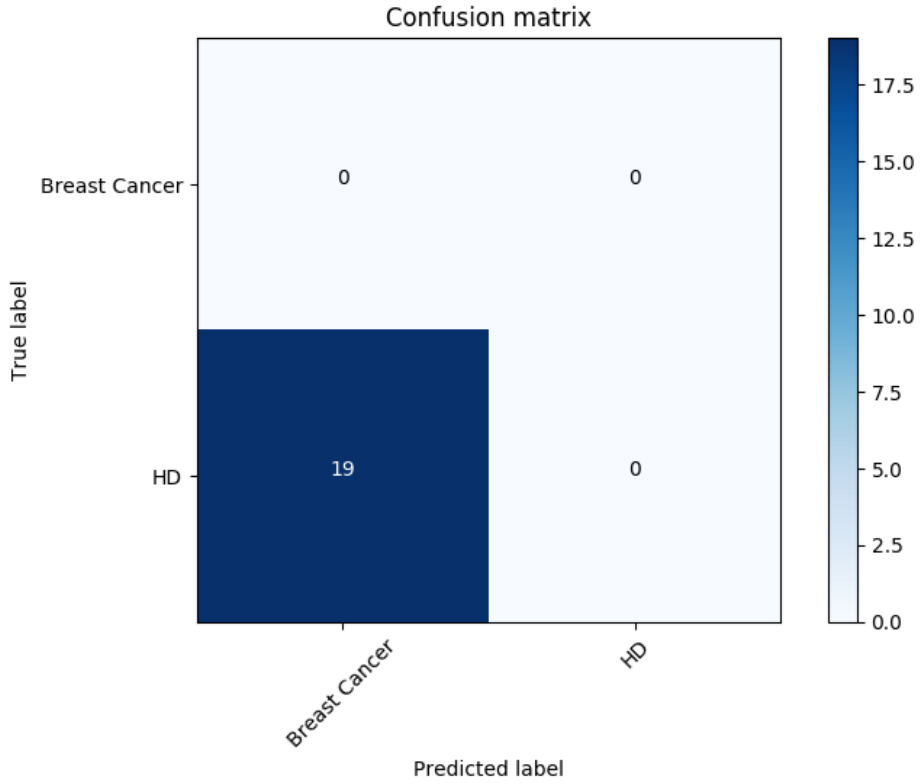

Figure 4 - Confusion matrix for evaluation of the classifier

E. Humanin Article List

Table 1 - Humanin article list and corresponding classification

Article Title	Classification
Humanin decreases mitochondrial membrane permeability by inhibiting the membrane association and oligomerization of Bax and Bid proteins.	relevant
Humanin is an endogenous activator of chaperone-mediated autophagy.	relevant
A Small Molecule Mimetic of the Humanin Peptide as a Candidate for Modulating NMDA-Induced Neurotoxicity.	relevant
Humanin affects object recognition and gliosis in short-term cuprizone-treated mice.	relevant
S14G-humanin alleviates insulin resistance and increases autophagy in neurons of APP/PS1 transgenic mouse.	relevant
Humanin analogue, S14G-humanin, has neuroprotective effects against oxygen glucose deprivation/reoxygenation by reactivating Jak2/Stat3 signaling through the PI3K/AKT pathway.	relevant
Pseudogenization of the Humanin gene is common in the mitochondrial DNA of many vertebrates.	relevant
Protective Mechanisms of the Mitochondrial-Derived Peptide Humanin in Oxidative and Endoplasmic Reticulum Stress in RPE Cells.	relevant
The Mitochondrial-Derived Peptides, HumaninS14G and Small Humanin-like Peptide 2, Exhibit Chaperone-like Activity.	relevant
Humanin Specifically Interacts with Amyloid-beta Oligomers and Counteracts Their in vivo Toxicity.	relevant
Endoplasmic reticulum-mitochondrial crosstalk: a novel role for the mitochondrial peptide humanin.	relevant
Serum humanin concentrations in women with pre-eclampsia compared to women with uncomplicated pregnancies.	relevant
Whole-transcriptome brain expression and exon-usage profiling in major depression and suicide: evidence for altered glial, endothelial and ATPase activity.	relevant
The mitochondrial-derived peptide humanin activates the ERK1/2, AKT, and STAT3 signaling pathways and has age-dependent signaling differences in the hippocampus.	relevant
Humanin: Functional Interfaces with IGF-I.	relevant
Central effects of humanin on hepatic triglyceride secretion.	relevant
The effects of humanin and its analogues on male germ cell apoptosis induced by chemotherapeutic drugs.	relevant
Humanin and age-related diseases: a new link?	relevant
Protection effect of [Gly14]-Humanin from apoptosis induced by high glucose in human umbilical vein endothelial cells.	relevant
Humanin attenuates Alzheimer-like cognitive deficits and pathological changes induced by amyloid beta-peptide in rats.	relevant
Protective effects of humanin on okadaic Acid-induced neurotoxicities in cultured cortical neurons.	relevant
Apollon/Bruce is upregulated by Humanin.	relevant
IGF-I regulates the age-dependent signaling peptide humanin.	relevant
Protective effects of Humanin and calmodulin-like skin protein in Alzheimer's disease and broad range of abnormalities.	relevant

Genome expression analysis by suppression subtractive hybridization identified overexpression of Humanin, a target gene in gastric cancer chemoresistance.	relevant
SH3-binding protein 5 mediates the neuroprotective effect of the secreted bioactive peptide humanin by inhibiting c-Jun NH2-terminal kinase.	relevant
Humanin Exerts Neuroprotection During Cardiac Ischemia-Reperfusion Injury.	relevant
Baculovirus-based gene silencing of Humanin for the treatment of pituitary tumors.	relevant
Calmodulin-like skin protein protects against spatial learning impairment in a mouse model of Alzheimer disease.	relevant
Humanin directly protects cardiac mitochondria against dysfunction initiated by oxidative stress by decreasing complex I activity.	relevant
Colivelin Ameliorates Impairments in Cognitive Behaviors and Synaptic Plasticity in APP/PS1 Transgenic Mice.	relevant
Humanin G (HNG) protects age-related macular degeneration (AMD) transmitochondrial ARPE-19 cybrids from mitochondrial and cellular damage.	relevant
Breaking the ritual metabolic cycle in order to save acetyl CoA: A potential role for mitochondrial humanin in T2 bladder cancer aggressiveness.	relevant
Humanin rescues cultured rat cortical neurons from NMDA-induced toxicity through the alleviation of mitochondrial dysfunction.	relevant
Humanin inhibits apoptosis in pituitary tumor cells through several signaling pathways including NF-kappaB activation.	relevant
Effects of humanin on experimental colitis induced by 2,4,6-trinitrobenzene sulphonic acid in rats.	relevant
Humanin skeletal muscle protein levels increase after resistance training in men with impaired glucose metabolism.	relevant
Humanin ameliorates diazepam-induced memory deficit in mice.	relevant
Humanin Protects RPE Cells from Endoplasmic Reticulum Stress-Induced Apoptosis by Upregulation of Mitochondrial Glutathione.	relevant
Rubimetide, humanin, and MMK1 exert anxiolytic-like activities via the formyl peptide receptor 2 in mice followed by the successive activation of DP1, A2A, and GABAA receptors.	relevant
Humanin exerts cardioprotection against cardiac ischemia/reperfusion injury through attenuation of mitochondrial dysfunction.	relevant
Solution NMR structure and inhibitory effect against amyloid-beta fibrillation of Humanin containing a d-isomerized serine residue.	relevant
Naturally occurring mitochondrial-derived peptides are age-dependent regulators of apoptosis, insulin sensitivity, and inflammatory markers.	relevant
Protective Effects of Colivelin Against Alzheimer's Disease in a PDAPP Mouse Model.	relevant
Potential Roles of Humanin on Apoptosis in the Heart.	relevant
The Potent Humanin Analogue (HNG) Protects Germ Cells and Leucocytes While Enhancing Chemotherapy-Induced Suppression of Cancer Metastases in Male Mice.	relevant
Humanin Peptide Binds to Insulin-Like Growth Factor-Binding Protein 3 (IGFBP3) and Regulates Its Interaction with Importin-beta.	relevant

Humanin Derivatives Inhibit Necrotic Cell Death in Neurons.	relevant
The human mitochondrial genome may code for more than 13 proteins.	relevant
New labeled derivatives of the neuroprotective peptide colivelin: synthesis, characterization, and first in vitro and in vivo applications.	relevant
Colivelin ameliorates amyloid beta peptide-induced impairments in spatial memory, synaptic plasticity, and calcium homeostasis in rats.	relevant
S14G-humanin restored cellular homeostasis disturbed by amyloid-beta protein.	relevant
Increased oligodendrogenesis by humanin promotes axonal remyelination and neurological recovery in hypoxic/ischemic brains.	relevant
Humanin rescues cultured rat cortical neurons from NMDA-induced toxicity not by NMDA receptor.	relevant
Potent humanin analog increases glucose-stimulated insulin secretion through enhanced metabolism in the beta cell.	relevant
A humanin analog decreases oxidative stress and preserves mitochondrial integrity in cardiac myoblasts.	relevant
Pharmacokinetics and tissue distribution of humanin and its analogues in male rodents.	relevant
Secreted calmodulin-like skin protein ameliorates scopolamine-induced memory impairment.	relevant
The cytoprotective peptide humanin is induced and neutralizes Bax after pro-apoptotic stress in the rat testis.	relevant
[Gly14]-Humanin offers neuroprotection through glycogen synthase kinase-3beta inhibition in a mouse model of intracerebral hemorrhage.	relevant
Low circulating levels of the mitochondrial-peptide hormone SHLP2: novel biomarker for prostate cancer risk.	non relevant
Subcellular Fractionation for ERK Activation Upon Mitochondrial-derived Peptide Treatment.	non relevant
High-dose Humanin analogue applied during ischemia exerts cardioprotection against ischemia/reperfusion injury by reducing mitochondrial dysfunction.	non relevant
Neuroprotective effect of G(14)-humanin on global cerebral ischemia/reperfusion by activation of SOCS3 - STAT3 - MCL-1 signal transduction pathway in rats.	non relevant
Calmodulin-like skin protein is downregulated in human cerebrospinal fluids of Alzheimer's disease patients with apolipoprotein E4; a pilot study using postmortem samples.	non relevant
Mitochondrially derived peptides as novel regulators of metabolism.	non relevant
Apoptotic neuron-secreted HN12 inhibits cell apoptosis in Hirschsprung's disease.	non relevant
Gly[14]-humanin inhibits ox-LDL uptake and stimulates cholesterol efflux in macrophage-derived foam cells.	non relevant
The Role of MicroRNAs and Their Targets in Osteoarthritis.	non relevant
Humanin: a mitochondrial signaling peptide as a biomarker for impaired fasting glucose-related oxidative stress.	non relevant

Exposure to sixty minutes of hyperoxia upregulates myocardial humanins in patients with coronary artery disease - a pilot study.	non relevant
A Fleeting Glimpse Inside microRNA, Epigenetics, and Micropeptidomics.	non relevant
Altered intestinal functions and increased local inflammation in insulin-resistant obese subjects: a gene-expression profile analysis.	non relevant
The mitochondrial-derived peptide MOTS-c: a player in exceptional longevity?	non relevant
Rat Humanin is encoded and translated in mitochondria and is localized to the mitochondrial compartment where it regulates ROS production.	non relevant
The mitochondrial-derived peptide MOTS-c promotes metabolic homeostasis and reduces obesity and insulin resistance.	non relevant
MTRNR2L12: A Candidate Blood Marker of Early Alzheimer's Disease-Like Dementia in Adults with Down Syndrome.	non relevant
Identification of Target Genes Regulated by KSHV miRNAs in KSHV-Infected Lymphoma Cells.	non relevant
The effect of sex on humanin levels in healthy adults and patients with uncomplicated type 1 diabetes mellitus.	non relevant
Antiapoptotic factor humanin is expressed in normal and tumoral pituitary cells and protects them from TNF-alpha-induced apoptosis.	non relevant
Potential peptides in atherosclerosis therapy.	non relevant
The neuroprotection of Rattin against amyloid beta peptide in spatial memory and synaptic plasticity of rats.	non relevant
Humanin: a novel functional molecule for the green synthesis of graphene.	non relevant
Distinct signaling cascades elicited by different formyl peptide receptor 2 (FPR2) agonists.	non relevant
Aeromedical solutions for aerospace safety.	non relevant

F. Practical Applications – Mindfulness Dataset Classification Reports

Table 2 - Classification report for the mindfulness dataset, for each training set and algorithm

Trial	Algorithm	Label	Precision	Recall	F1-Score	Support	Accuracy*	Conf. Matrix
Train	Multinomial NB	SRincluded	0%	0%	0%	4	90,75%	[0 4] [7 108]
		mindf_fatigue	96%	94%	95%	115		
		avg / total	93%	91%	92%	119		
	K Neighbors	SRincluded	7%	25%	11%	4	85,71%	[1 3] [14 101]
		mindf_fatigue	97%	88%	92%	115		
		avg / total	94%	86%	89%	119		
	Random Forest	SRincluded	0%	0%	0%	4	89,08%	[0 4]

		mindf_fatigue	96%	92%	94%	115		[9 106]
		avg / total	93%	89%	91%	119		
	Decision Trees	SRincluded	8%	25%	12%	4	88,24%	[1 3] [11 104]
		mindf_fatigue	97%	90%	94%	115		
		avg / total	94%	88%	91%	119		
	Logistic Regression	SRincluded	12%	25%	17%	4	91,60%	[1 3] [7 108]
		mindf_fatigue	97%	94%	96%	115		
		avg / total	94%	92%	93%	119		
Train 1	Multinomial NB	SRincluded	9%	50%	15%	4	80,67%	[2 2] [21 94]
		mindf_fatigue	98%	82%	89%	115		
		avg / total	95%	81%	87%	119		
	K Neighbors	SRincluded	11%	50%	17%	4	84,03%	[2 2] [17 98]
		mindf_fatigue	98%	85%	91%	115		
		avg / total	95%	84%	89%	119		
	Random Forest	SRincluded	5%	50%	9%	4	69,70%	[2 2] [40 75]
		mindf_fatigue	97%	65%	78%	115		
		avg / total	94%	65%	76%	119		
	Decision Trees	SRincluded	10%	75%	18%	4	76,47%	[3 1] [27 88]
		mindf_fatigue	99%	77%	86%	115		
		avg / total	96%	76%	84%	119		
	Logistic Regression	SRincluded	9%	50%	15%	4	80,67%	[2 2] [21 94]
		mindf_fatigue	98%	82%	89%	115		
		avg / total	95%	81%	87%	119		
Train 2	Multinomial NB	SRincluded	5%	100%	10%	4	36,97%	[4 0] [75 40]
		mindf_fatigue	100%	35%	52%	115		
		avg / total	97%	37%	50%	119		
	K Neighbors	SRincluded	5%	100%	10%	4	39,50%	[4 0] [72 43]
		mindf_fatigue	100%	37%	54%	115		
		avg / total	97%	39%	53%	119		
	Random Forest	SRincluded	4%	50%	7%	4	52,94%	[2 2] [54 61]
		mindf_fatigue	97%	53%	69%	115		
		avg / total	94%	53%	66%	119		
	Decision Trees	SRincluded	6%	75%	11%	4	57,14%	[3 1] [50 65]
		mindf_fatigue	98%	57%	72%	115		
		avg / total	95%	57%	70%	119		
	Logistic Regression	SRincluded	5%	75%	10%	4	53,78%	[3 1] [54 61]
		mindf_fatigue	98%	53%	69%	115		
		avg / total	95%	54%	67%	119		
Train 3	Multinomial NB	SRincluded	6%	50%	11%	4	72,27%	[2 2] [31 84]
		mindf_fatigue	98%	73%	84%	115		

		avg / total	95%	72%	81%	119		
	K Neighbors	SRincluded	6%	75%	11%	4	57,98%	[3 1] [49 66]
		mindf_fatigue	99%	57%	73%	115		
		avg / total	95%	58%	70%	119		
	Random Forest	SRincluded	13%	75%	22%	4	82,35%	[3 1] [20 95]
		mindf_fatigue	99%	83%	90%	115		
		avg / total	96%	82%	88%	119		
	Decision Trees	SRincluded	12%	25%	17%	4	91,60%	[1 3] [7 108]
		mindf_fatigue	97%	94%	96%	115		
		avg / total	94%	92%	93%	119		
	Logistic Regression	SRincluded	9%	50%	15%	4	80,67%	[2 2] [21 94]
		mindf_fatigue	98%	82%	89%	115		
		avg / total	95%	81%	87%	119		

G. Practical Applications – Humanin Dataset Classification Reports

Table 3 - Classification report for the humanin dataset, for each training set and algorithm

Trial	Algorithm	Label	Precision	Recall	F1-Score	Support	Accuracy	Conf. Matrix
Train 1	Multinomial NB	nonrelevant	44%	56%	49%	25	65,88%	[14 11] [18 42]
		relevant	79%	70%	74%	60		
		avg / total	69%	66%	67%	85		
	K Neighbors	nonrelevant	48%	48%	48%	25	69,41%	[12 13] [13 47]
		relevant	78%	78%	78%	60		
		avg / total	69%	69%	69%	85		
	Random Forest	nonrelevant	48%	48%	48%	25	69,41%	[12 13] [13 47]
		relevant	78%	78%	78%	60		
		avg / total	69%	69%	69%	85		
	Decision Trees	nonrelevant	47%	28%	35%	25	69,41%	[7 18] [8 52]
		relevant	74%	87%	80%	60		
		avg / total	66%	69%	67%	85		
	Logistic Regression	nonrelevant	53%	68%	60%	25	72,94%	[17 8] [15 45]
		relevant	85%	75%	80%	60		
		avg / total	76%	73%	74%	85		
Train 2	Multinomial NB	nonrelevant	54%	60%	57%	25	72,94%	[15 10] [13 47]
		relevant	82%	78%	80%	60		
		avg / total	74%	73%	73%	85		
	K Neighbors	nonrelevant	54%	60%	57%	25	72,94%	[15 10] [13 47]
		relevant	82%	78%	80%	60		

		avg / total	74%	73%	73%	85		
	Random Forest	nonrelevant	46%	84%	59%	25	65,88%	[21 4] [25 35]
		relevant	90%	58%	71%	60		
		avg / total	77%	66%	67%	85		
	Decision Trees	nonrelevant	32%	32%	32%	25	60%	[8 17] [17 43]
		relevant	72%	72%	72%	60		
		avg / total	60%	60%	60%	85		
	Logistic Regression	nonrelevant	58%	60%	59%	25	75,29%	[15 10] [11 49]
		relevant	83%	82%	82%	60		
		avg / total	76%	75%	75%	85		
Train 3	Multinomial NB	nonrelevant	52%	48%	50%	25	71,76%	[12 13] [11 49]
		relevant	79%	82%	80%	60		
		avg / total	71%	72%	71%	85		
	K Neighbors	nonrelevant	55%	44%	49%	25	72,94%	[11 14] [9 51]
		relevant	78%	85%	82%	60		
		avg / total	72%	73%	72%	85		
	Random Forest	nonrelevant	71%	20%	31%	25	74,12%	[5 20] [2 58]
		relevant	74%	97%	84%	60		
		avg / total	73%	74%	69%	85		
	Decision Trees	nonrelevant	25%	4%	7%	25	68,24%	[1 24] [3 57]
		relevant	70%	95%	81%	60		
		avg / total	57%	68%	59%	85		
	Logistic Regression	nonrelevant	75%	48%	59%	25	80%	[12 13] [4 56]
		relevant	81%	93%	87%	60		
		avg / total	79%	80%	79%	85		
Train 4	Multinomial NB	nonrelevant	61%	44%	51%	25	75,29%	[11 14] [7 53]
		relevant	79%	88%	83%	60		
		avg / total	74%	75%	74%	85		
	K Neighbors	nonrelevant	47%	32%	38%	25	69,41%	[8 17] [9 51]
		relevant	75%	85%	80%	60		
		avg / total	67%	69%	67%	85		
	Random Forest	nonrelevant	60%	12%	20%	25	71,76%	[3 22] [2 58]
		relevant	72%	97%	83%	60		
		avg / total	69%	72%	64%	85		
	Decision Trees	nonrelevant	33%	4%	7%	25	69,41%	[1 24] [2 58]
		relevant	71%	97%	82%	60		
		avg / total	60%	69%	60%	85		
	Logistic Regression	nonrelevant	80%	48%	60%	25	81,17%	[12 13] [3 57]
		relevant	81%	95%	88%	60		
		avg / total	81%	81%	80%	85		

Train 5	Multinomial NB	nonrelevant	61%	44%	51%	25	75,29%	[11 14] [7 53]
		relevant	79%	88%	83%	60		
		avg / total	74%	75%	74%	85		
	K Neighbors	nonrelevant	47%	32%	38%	25	69,41%	[8 17] [9 51]
		relevant	75%	85%	80%	60		
		avg / total	67%	69%	67%	85		
	Random Forest	nonrelevant	40%	8%	13%	25	69,41%	[2 23] [3 57]
		relevant	71%	95%	81%	60		
		avg / total	62%	69%	61%	85		
	Decision Trees	nonrelevant	33%	4%	7%	25	69,41%	[1 24] [2 58]
		relevant	71%	97%	82%	60		
		avg / total	60%	69%	60%	85		
	Logistic Regression	nonrelevant	77%	40%	53%	25	78,82%	[10 15] [3 57]
		relevant	79%	95%	86%	60		
		avg / total	79%	79%	76%	85		
Train 6	Multinomial NB	nonrelevant	55%	44%	49%	25	72,94%	[11 14] [9 51]
		relevant	78%	85%	82%	60		
		avg / total	72%	73%	72%	85		
	K Neighbors	nonrelevant	53%	40%	45%	25	71,76%	[10 15] [9 51]
		relevant	77%	85%	81%	60		
		avg / total	70%	72%	71%	85		
	Random Forest	nonrelevant	60%	12%	20%	25	71,76%	[3 22] [2 58]
		relevant	72%	97%	83%	60		
		avg / total	69%	72%	64%	85		
	Decision Trees	nonrelevant	33%	4%	7%	25	69,41%	[1 24] [2 58]
		relevant	71%	97%	82%	60		
		avg / total	60%	69%	60%	85		
	Logistic Regression	nonrelevant	73%	44%	55%	25	78,82%	[11 14] [4 56]
		relevant	80%	93%	86%	60		
		avg / total	78%	79%	77%	85		
Train 7	Multinomial NB	nonrelevant	54%	52%	53%	25	72,94%	[13 12] [11 49]
		relevant	80%	82%	81%	60		
		avg / total	73%	73%	73%	85		
	K Neighbors	nonrelevant	60%	48%	53%	25	75,29%	[12 13] [8 52]
		relevant	80%	87%	83%	60		
		avg / total	74%	75%	74%	85		
	Random Forest	nonrelevant	60%	24%	34%	25	72,94%	[6 19] [4 56]
		relevant	75%	93%	83%	60		
		avg / total	70%	73%	69%	85		
	Decision Trees	nonrelevant	43%	12%	19%	25	69,41%	[3 22]

		relevant	72%	93%	81%	60		[4 56]
		avg / total	63%	69%	63%	85		
	Logistic Regression	nonrelevant	75%	48%	59%	25	80%	[12 13]
		relevant	81%	93%	87%	60		[4 56]
		avg / total	79%	80%	79%	85		
Train 8	Multinomial NB	nonrelevant	53%	40%	45%	25	71,76%	[10 15]
		relevant	77%	85%	81%	60		[9 51]
		avg / total	70%	72%	71%	85		
	K Neighbors	nonrelevant	52%	44%	48%	25	71,76%	[11 14]
		relevant	78%	83%	81%	60		[10 50]
		avg / total	71%	72%	71%	85		
	Random Forest	nonrelevant	50%	16%	24%	25	70,59%	[4 21]
		relevant	73%	93%	82%	60		[4 56]
		avg / total	66%	71%	65%	85		
	Decision Trees	nonrelevant	62%	20%	30%	25	72,94%	[5 20]
		relevant	74%	95%	83%	60		[3 57]
		avg / total	71%	73%	68%	85		
	Logistic Regression	nonrelevant	65%	44%	52%	25	76,47%	[11 14]
		relevant	79%	90%	84%	60		[6 54]
		avg / total	75%	76%	75%	85		
Train 9	Multinomial NB	nonrelevant	54%	52%	53%	25	72,94%	[13 12]
		relevant	80%	82%	81%	60		[11 49]
		avg / total	73%	73%	73%	85		
	K Neighbors	nonrelevant	57%	52%	54%	25	74,11%	[13 12]
		relevant	81%	83%	82%	60		[10 50]
		avg / total	74%	74%	74%	85		
	Random Forest	nonrelevant	50%	20%	29%	25	70,59%	[5 20]
		relevant	73%	92%	81%	60		[5 55]
		avg / total	66%	71%	66%	85		
	Decision Trees	nonrelevant	50%	16%	24%	25	70,59%	[4 21]
		relevant	73%	93%	82%	60		[4 56]
		avg / total	66%	71%	65%	85		
	Logistic Regression	nonrelevant	75%	48%	59%	25	80,00%	[12 13]
		relevant	81%	93%	87%	60		[4 56]
		avg / total	79%	80%	79%	85		
Train 10	Multinomial NB	nonrelevant	52%	44%	48%	25	71,76%	[11 14]
		relevant	78%	83%	81%	60		[10 50]
		avg / total	71%	72%	71%	85		
	K Neighbors	nonrelevant	55%	48%	51%	25	72,94%	[12 13]
		relevant	79%	83%	81%	60		[10 50]

		avg / total	72%	73%	72%	85		
	Random Forest	nonrelevant	55%	24%	33%	25	71,76%	[6 19] [5 55]
		relevant	74%	92%	82%	60		
		avg / total	69%	72%	68%	85		
	Decision Trees	nonrelevant	62%	20%	30%	25	72,94%	[5 20] [3 57]
		relevant	74%	95%	83%	60		
		avg / total	71%	73%	68%	85		
	Logistic Regression	nonrelevant	61%	44%	51%	25	75,29%	[11 14] [7 53]
		relevant	79%	88%	83%	60		
		avg / total	74%	75%	74%	85		
Train 11	Multinomial NB	nonrelevant	100%	32%	48%	25	80,00%	[8 17] [0 60]
		relevant	78%	100%	88%	60		
		avg / total	84%	80%	76%	85		
	K Neighbors	nonrelevant	56%	20%	29%	25	71,76%	[5 20] [4 56]
		relevant	74%	93%	82%	60		
		avg / total	68%	72%	67%	85		
	Random Forest	nonrelevant	56%	20%	29%	25	71,76%	[5 20] [4 56]
		relevant	74%	93%	82%	60		
		avg / total	68%	72%	67%	85		
	Decision Trees	nonrelevant	67%	8%	14%	25	71,76%	[2 23] [1 59]
		relevant	72%	98%	83%	60		
		avg / total	70%	72%	63%	85		
	Logistic Regression	nonrelevant	79%	44%	56%	25	80,00%	[11 14] [3 57]
		relevant	80%	95%	87%	60		
		avg / total	80%	80%	78%	85		