Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Machine learning for disambiguation of clinical trial scientist names

Studente/i

Matteo Bresciani

Relatore

Fabio Rinaldi

Correlatore

Luca Maria Gambardella

Committente

Hoffmann-La Roche

Corso di laurea

Ingegneria Informatica

Modulo

Progetto di Diploma

Anno

2018/2019

Data

August 30, 2019

Contents

Chapter 1

Abstract

Italiano

L'AND (noto come author name disambiguation) è un problema noto, non essendo diffuso l'uso di identificatori univoci per gli autori, è quindi difficile discernere se due autori con lo stesso cognome e con la stessa iniziale del nome siano la stessa persona o meno.

In passato sono stati creati molti algoritmi per cercare di risolvere questo problema, gli algoritmi creati non risolvono però ancora il problema con sufficiente precisione.

Gran parte di essi è stata realizzata confrontando solo coppie di articoli presi da MEDLINE. Lo scopo del mio lavoro è quello di creare algoritmi in grado di risolvere il problema dell'AND con sufficiente precisione, ma confrontando gli autori degli articoli di MEDLINE con quelli degli studi clinici.

Come vedremo, il confronto dà come risultato una precisione paragonabile ai precedenti algoritmi realizzati confrontando articoli provenienti esclusivamente da MEDLINE.

English

Author name disambiguation (AND) is a known problem. There is no widespread use of unique identifiers for the authors, thus it is difficult to discern whether or not two authors with the same last name and with the same first name initial are the same person.

In the past, there have been many attempts to solve this problem. The algorithms that were created do not still solve the problem with enough precision.

The vast majority of them have been made comparing only pairs of articles taken from the MEDLINE library. The purpose of my work is to make an algorithm that can solve the AND problem with enough precision, but comparing authors from MEDLINE articles with authors from clinical trials.

As we will see, the comparison yields a precision comparable to the previous algorithms made comparing articles from the same library.

Chapter 2

Introduction

2.1 Description of the project

When searching for scientific literature, author names are the most frequent requests and the use of their abbreviation in queries is also common [1], the problem is that most scientific publications cannot be associated to a specific author.

Author name disambiguation(AND) then becomes a crucial problem to solve in order to correctly address the authors to their publications.
Many authors of scientific literature share the same last name and the same first name initial and that complicates the job of disambiguation of the authors.
To solve this problem, a number of solutions have been proposed throughout the years, like an unique identifier for the author, such as ORCID, but their use is not yet widespread.

Other solutions attempt to solve this problem using machine-learning algorithms, like the one that we are proposing, and they have been able to achieve high degrees of precision. The disambiguation process involves a pair of articles whom authors share the same namespace [1].

The related work has been done trying to disambiguate the authors in pairs composed by two PubMed articles. I am trying to disambiguate the authors in pairs composed by a PubMed article and a clinical trial.

[1] a namespace comprises all the authors with the same last name and the same first name initial

Chapter 3

Related Works

3.1 Previous works on the subject

To solve the AND problems, many machine-learning algorithms have been created.
They can be divided in supervised and unsupervised.

Regarding the clustering (or unsupervised) algorithms, they do not need a gold standard to be created. They try to separate the authors in entities, having every entity to represent a physical person.

The machine learning algorithms using supervised learning have achieved greater results than the unsupervised ones.
These algorithms need a gold standard. A problem with that is having a non-representative gold standard and this is sometimes the case. Other problems can consist in not using articles that have missing information in them.
Not having a representative gold standard can lead to having better results, but the results will not be usable because they do not represent the database in which they have been taken.

A solution to this is using an unbiased gold standard that is able to represent the whole database (MEDLINE and ClinicalTrials.gov in our case, both curated by United States National Library of Medicine).

A biased gold standard can have an over representation of English authors [2] or it can discard articles based on information presence, like the Arnetminer corpus [3] or it can select the authors based on name popularity [4].

The gold standard we are going to use was made from ROCHE purposely for this project, it also takes into consideration the missing information.

Usually, algorithms that aims to solve the author name disambiguation problem base their disambiguation methods on an author personal data, like name, affiliation, e-mail and co-authors. This poses a problem because this information is not always provided in articles from MEDLINE.

To overcome this problem, some text-categorisation tools have been used in some researches, like the ones done by ROCHE, as we will see in the next section.

3.2 Related works done by ROCHE

In this section I describe the previous work done by ROCHE on AND and the main features they have used.

They have done two projects in this regards, the first one is called "Author Name Disambiguation in MEDLINE Based on Journal" [5] and the second is called "A new approach and gold standard toward author name disambiguation in MEDLINE" [6].

Both of them introduce new concepts in the solution of the AND problem. The first introduces the Text Categorisation with journal descriptors and semantic types and the second introduces an unbiased gold standard.

3.2.1 Author Name Disambiguation in MEDLINE Based on Journal Descriptors and Semantic Types

The paper introduces the problem of Author name disambiguation (AND) and it explains that emails and affiliations of the authors are usually missing in the articles found in Pubmed.

This paper also inspired me to use the Stanford NER to extract information like cities, countries and organizations from texts, here they use it to recognise the organizations.

They used journal descriptors (JD) and semantic types (ST) as new features to solve this problem and achieved good performances, founding that these features had the same impact (if not greater) than other features like email addresses.

They used these attributes:

- Organization
- Full First Name
- Email
- Type and Descriptor of Organization
- Coauthors
- Journal descriptors
- Semantic types
- City
- Year
- Initials
- Country
- Language
- Mesh Terms

In this research they fund out that MeSH Terms are not a lot useful when it comes to solving the author name disambiguation problem, in fact, they had 0.0 as feature importance for these terms.

To classify the authors, they used J48, a random forest classifier, a k-NN classifier, and an SVM one. They found that random forest is far better than SVM to solve this type of problems (I share these results in my project, in the section "Results").

This work was done by Dina Vishnyakova, Raul Rodriguez-Esteban, Khan Ozol and Fabio Rinaldi.

3.2.2 A new approach and gold standard toward author name disambiguation in MEDLINE

This work differs from the one listed above because here we have a gold standard.
The gold standard in question was a document that contained rows of article-article pairs, the namespace (last name and first name initial) of an author present in both articles and the verdict on whether or not, for the experts, the author was the same person.

Before using the experts, the authors of this paper tried to use crowd-sourcing platforms to create the gold standard.
In some cases this can work, but in the specific case of author name disambiguation, it has not worked. Using Figure Eight and Amazon Mechanical Turk they have not managed to get high accuracy responses, given this fact, crowd-sourcing had to be abandoned for the project and expert curators were adopted.

The attributes taken into consideration here are the same as the previous research listed before (no Mesh Terms in this one), plus these:

- Ambiguity score
- Last name length

They compared the results they got with their gold standard (an unbiased one) with the gold standard used in SONG. Results were better if trained and tested on SONG's gold standard, but they were also not representative of the MEDLINE library. Using the model trained on SONG and tested on their data-set. They obtained poor results, that because SONG is a biased gold standard, when training on their gold standard and testing on SONG, results have improved a lot, because the gold standard they have created is a lot more representative of the data-set.

This work was done by Dina Vishnyakova, Raul Rodriguez-Esteban and Fabio Rinaldi.

Chapter 4

Data

4.1 Available data

The data available for the project consist in:

- Gold standard
- Pubmed articles
- Clinical trials

4.1.1 Gold standard

The gold standard is a document that contains 1162 rows.
Every row contains the namespace of the author (last name and first name initial), the ID of the article and the ID of the clinical trial, articles are paired with clinical trials. After these two IDs and the namespace, we get the verdict of each of the three experts that says whether the author is the same person in the clinical trial and in the article or if he/she is not.
We also have the common answer, that is the average of the answers of the experts.

4.1.2 Pubmed articles

Pubmed is an engine that is used to access MEDLINE database.
This database is maintained by the National Library of Medicine and contains scientific literature based on topics regarding life sciences and biomedical.

Having only the gold standard as resource, I made requests online in order to have access to all the articles I required, I have also done a small algorithm capable of saving the articles on a folder that I used later, this process reduced the computational time because we no longer needed to always make requests to get them.

These articles contain information about the abstract, the authors involved into the research, their affiliations and so on. We extracted information from these articles and used them in combination with the information from the clinical trials in order to find useful attributes that could be used to train a classifier.

4.1.3 Clinical trials

The clinical trials were provided by the site clinicaltrials.gov, the database is curated by the U.S. National Library of Medicine.

Compared to the articles, the clinical trials are few, so, instead of making requests to the site, I simply downloaded all of them and placed them in the folder 'AllPublicXML', we can easily access them offline.

In the clinical trials we can find a lot of useful information that can be used later for the purpose of training our classifier.

Chapter 5

Methodologies

5.1 Overview

Here I want to give you an overview of all the components of the project.

The components are:

- Data frame creator
- Model creator

5.1.1 Data frame creator

The data frame creator is used to create a data frame that contains all the attributes that will be used in the classifier.

Simply put, in this component I will extract all the information that I need from the articles and from the clinical trials.

All the information extracted has numerical values. I will explain later the calculations done to make this possible. The data frame with all the calculated values is then saved in a csv file called "dataframe.csv" and then it will be used by the classifier.

5.1.2 Model creator

This component is responsible for the training and the testing of the data frame created in the component listed above.

The model created with this procedure is then saved into the "models" folder and can be used later to classify other pairs of article-clinical trial.

Using the features, the classifier has to assess whether or not the author in the article and the author in the clinical trial is the same person.

5.2 Features

In this section I will describe the features used in the classifier, the features are the following:

- Full First Name
- Organization
- Email
- Type of Organization
- City
- Country
- Year difference
- Initials
- Ambiguity score
- Last name length
- Journal descriptors
- Semantic types
- Doc2vec
- Oger

5.2.1 Full First Name

This feature compares the full first names of the authors of the clinical trial and of the article, the comparison is made using the levenshtein comparison between the two strings containing the names and then I divide the result by the max length of the two strings.
The formula is:

$$1 - (levenshtein(first\_name1, first\_name2)/max(len(first\_name1), len(first\_name2)))$$

5.2.2 Organization

I use three different ways to extract the affiliation of the authors:

- Direct
- Extraction with Spacy
- Extraction with Stanford

5.2.2.1 Direct

The output of this method consists in one string containing the content of the affiliation tag in the XML file.

The Direct approach consists in finding in the XML file (clinical trial or article) the affiliation of the author. In the articles, there is a list of all the authors and affiliation. I find the affiliation in the author tag.
For the clinical trials, the author affiliation can be found in different parts of the XML. That is because information regarding the author are in different places. Once found where the author name is, I can find the affiliation if present.

The organization of the author is usually missing in the articles, but almost always present in the clinical trials.

The Direct approach is used to find all the affiliation tags. In the clinical trials the tags are clean, only containing the organization name and in some cases, the country and the city. For the articles, the affiliation tag usually contains way more than that, sometimes even other authors. Using the 2 other approaches, I can recognise the organizations inside of the tags.

The direct approach is always needed to find the content of the affiliation tags, the other two methods extract information from the result of the direct approach.

5.2.2.2 Extraction with Spacy

This method takes the output of the direct method as input and uses Spacy as a library in order to extract the affiliation from the text contained in the affiliation tag.

The result is an array that contains all the affiliations that Spacy has found.

5.2.2.3 Extraction with Stanford

The Stanford NER is one of the most used methods when it comes to entity name recognition. The procedure is almost the same as with Spacy, it takes the text of the affiliation tag as input and it returns an array containing all the affiliations that it has recognised.

The only difference between these two methods (apart from the library used), is that the Stanford NER library needs to be run using java.

It takes the java path specified in the file "java_path.txt" and it uses it in order to run the library.

5.2.2.4 Comparison between the organizations

If one or both the organization are missing, the similarity of the organizations is 0. If both the organizations are extracted using the direct method, we have two strings to compare and the formula used to assess their similarity score is:

$$1 - (levenshtein(org1, org2)/max(len(org1), len(org2)))$$

If we use the Spacy/Stanford methods to get the organization of the articles and/or the clinical trials, we will have 2 arrays of possible organizations (in case we use the Spacy/Stanford method only on clinical trials or only on articles, we would have to compare an array and a string, I converted the string into an array of strings of length 1, so we will be comparing two arrays of strings). The method used to compare them is similar to the one explained above, but it is done for every pair of string present in the array.

The result of these comparisons will be the highest value obtained comparing all the possible pairs of strings between the two arrays.

Sometimes in an organization, there are 1-2 words not related to the name of the organization itself (like the abbreviation of the city or the ISO name of the country), so, if the score of similarity is equal or greater than 0.9, I will set it to 1.

5.2.3 E-mail

This feature is a comparison between the e-mail found in the article and the one found in the clinical trial, this feature is either zero or one.

Speaking about the articles, the e-mail is found in the affiliation tag, I search for the character "@" and I extract the surrounding e-mail. In the clinical trials, the e-mail is found in different locations, it can be found in "point_of_contact" or in an "e-mail" tag in the author tag.

Unfortunately, this information is not present most of the times, and the usefulness of this feature is limited.

5.2.4 Type of Organization

This features simply try to assess whether the type of the organization of the affiliations of the 2 authors is the same or not, the possible types of organization considered are:

- University
- School
- Institute
- Hospital

The name of the organization taken into consideration is the one extracted using one of the three methods of affiliation extraction listed above.

If the two strings (or at least two strings contained in the 2 arrays, in case of at least one extraction using Spacy/Stanford extraction techniques) contain the same type of organization (E.g. I check whether the word "University" is present in both organizations' strings), I assign 1, 0 otherwise.

Even though sometimes I do not find the same type of the organization, If the organization similarity score (described above) is 1, I also set the type of organization to 1 because if the name is exactly the same, the type of the organization cannot be different.

5.2.5 City

This feature compares the cities found in the clinical trials with the ones found in the articles.

The cities in the clinical trials are found in tags called "location", I take all the cities in the tags and I store them in an array. For the article is different, the city (if present) is found in the affiliation tag and it regards the city of the affiliation of the author, I still use the same three methods to get the city from the tag: Direct, Spacy and Stanford.

The direct methods simply give you an array of all the words in the affiliation tag, the Spacy/Stanford methods, searches in the tag and they also give you an array of all the cities and of countries found using their libraries.

Once I have the 2 arrays set up, I search them in order to find out if they have cities in common, If one city in the array of the cities of the clinical trial is equal to one city in the array of the article, I set this feature to 1, 0 otherwise.

5.2.6 Country

This feature does the same thing that the "City" feature does, the only difference is that I compare the countries and not the cities.

The countries in the clinical trials are found in the location attribute described above. For the articles, I use the same array I used for the cities, because it also contains the countries.

Now that I have the 2 arrays, I search for equalities in the countries, using the same methods I used for the cities, but comparing countries. If I find a match, I place 1, 0 otherwise.

Sometimes I see USA or UK instead of the full names, I convert those names into the complete ones so that I do not miss potential matches.

5.2.7 Year difference

This feature represents the difference in years between the publications of the clinical trial and the article.

In the clinical trials, the year is taken from the field "study_first_submitted". In the articles the date is taken from the field PubDate or (if this field is missing) from the field "MedlineDate".

The formula used is simply:

$$abs(year1 - year2)$$

The results are normalised in the range 0.0 - 1.0.

5.2.8 Initials

This feature compares the initials of the names of the two authors.

Once the name is extrapolated, I remove all the special characters from the name and I take all the initials of all the names.

This feature can only be 0 (if the initials are different) or 1 (if they match).

5.2.9 Ambiguity score

This feature represents the namespace sizes of the namespaces in which the two authors belong.

Given the fact that the authors have to belong to the same namespace to be in a clinical trial-article pair, I calculate the namespace size based on the namespace of the two authors.

I get the namespace sizes by making a GET request to E-Utilities, the request is made in order to get all the publications in MEDLINE in which a certain author is present.

In the request I specify the author last name and the author first initial. When I get the result, I count the numbers of the publications, that is the namespace size (or as I call it here, the ambiguity score).

The results are then normalised in the range 0.0 - 1.0.

5.2.10 Last name length

This feature represents the length of the last name of the authors.

A big proportion of the most ambiguous author names is from Asian countries, and it is of a relatively short length, so this feature helps recognising the namespaces that can be more ambiguous than others.

Once I have all the last name lengths, I normalise them in the interval 0.0 - 1.0.

5.2.11 JDIs

With this feature (and some others that I will explain next), the goal is to assess the similarity of the article and of the clinical trial, based on the content of the title and the content of the text.

This feature is obtained comparing the journal descriptors extracted from the article and from the clinical trial.

5.2.11.1 Extracting the Journal Descriptors

The library that is able to extract these journal descriptors cannot be used in python, so we will need to use it in java.

I downloaded the library called "tc2011", this is the library that enables me to extract the information that I need. I created a java program that interfaces with their API and creates a Gateway server that can be interrogated by a python program.

The java program receives a text as input and return a string that contains all the journal descriptors and the confidence of it. I take the first 100 journal descriptor, their are in order of confidence, so the first 100 represent an accurate guess. The java program interrogates the API of the "tc2011" library in order to return the journal descriptors.

To be used correctly from python, the java program needs to be accessible from it, so I created a Gateway server from it, the server can receive requests for the semantic types and for the journal descriptors. Once set up, the server is able to receive requests from python and to return the desired result.

Speaking about the python program, it makes requests to the server described above, once it gets the result. It filters it in order to get the 100 journal descriptors.

The output of the filtering is an array of 100 elements structures as (confidence of the journal descriptor, name of journal descriptor). The confidence of the journal descriptor can range from 0.0 to 1.0.

5.2.11.2 Using the Journal Descriptor

Now that we have a way to get the journal descriptors, I get them for the clinical trials and for the articles. I can set how many journal descriptors or how many semantic types I want (1-100), this changes the result a little bit in the classifier.

Now we need to compare them, I have four ways to compare the journal descriptors and the semantic types:

- Basic
- Confidence
- Ranking
- Confidence Ranking

5.2.11.3 Basic Comparison

This comparison simply counts the number of journal descriptors that the clinical trials and the articles have in common and then it divides the result by the maximum number of journal descriptor present (1-100, it can be set in the dataframe_creator file). The formula is the following:

$$common\_journal\_descriptor(article\_JDIs, clinical\_trial\_JDIs)/maximum\_journal\_descriptors$$

5.2.11.4 Confidence Comparison

This comparison takes into consideration the confidence of the journal descriptors.

I sum all the shared parts of confidence of the common journal descriptors of articles and clinical trials (E.g. if I have a journal descriptor on the article (0.25 | Radiology) and another one on the clinical trial (0.20 | Radiology), I add 0.20 to the sum) and then I divide them by the maximum between the sum of all the confidences of all the journal descriptors of the article and the sum of all the confidences of all the journal descriptors of the clinical trial.

Here there is the pseudo code of the procedure:

```
jdis_sum(jdis):
    sum = 0
    for confidence in jdis:
        sum += confidence
    return sum

confidence_comparison(ct_jdis, ar_jdis):
    max_confidence = max(jdis_sum(ct_jdis), jdis_sum(ar_jdis))
    confidence_sum = 0
    for confidence_ct, ct_term in ct_jdis:
        for confidence_ar, ar_term in ar_jdis:
            if term_1 == term_2:
                current_max = max(confidence_ar, confidence_ct)
                absolute_difference = abs(confidence_ct - confidence_ar)
                confidence_sum += (current_max - absolute_difference)
    return confidence_sum/max_confidence
```

5.2.11.5 Ranking Comparison

This comparison takes into consideration the positions of the journal descriptor in the array of journal descriptors. The higher the descriptor, the higher the confidence that the text belongs to it, the higher the weight that I place on it.

The value of a journal descriptor is defined by 1 / ranking position (E.g. for a journal descriptor in position 3, the value will be 1/3) . The total value of the similarity is obtained summing all the multiplications of the matching journal descriptor values. After doing that, I divide the result by the maximum possible score.

Here there is the pseudo code of the procedure:

```
max_score(num):
    sum = 0
    for i 1...num:
        similarity += (num/(i))^2
    return similarity

ranking_comparison(ct_jdis, ar_jdis):
    length = ct_jdis.length    //both have the same length
    max_similarity = max_score(length)
    similarity = 0
    for i 1...length:
        for j 1...length:
            if ct_jdis[i].term == ar_jdis[j].term:
                similarity += 1/i * 1/j
    return similarity/max_similarity
```

5.2.11.6 Confidence Ranking Comparison

This comparison aims to be a mix between the confidence comparison and the ranking comparison. It takes into consideration the confidence and the ranking of the journal descriptor.

It is very similar to the confidence comparison, but it also takes into account the position of the journal descriptor in the array. The confidence score is calculated in the exact same way of the confidence comparison, but then it is multiplied by the weight of the journal descriptor (the weight is calculated in the same way as the ranking comparison), then everything is divided by the maximum possible result achievable.

Here there is the pseudo code of the procedure:

```
max_sum(jdis):
    length = jdis.length
    confidence_sum = 0
    for i 1...length:
        confidence_sum += (jdis[i].confidence^2) / (i^2)
    return confidence_sum


confidence_ranking_comparison(ct_jdis, ar_jdis):
    max_confidence_sum = max(max_sum(ct_jdis), max_sum(ar_jdis))
    confidence_sum = 0
    for i 1...length:
        for j 1...length:
            if ct_jdis[i].term == ar_jdis[j].term:
                current_max = max(ct_jdis[i].conf, ar_jdis[j].conf)
                absolute_difference = abs(ct_jdis[i].conf - ar_jdis[j].conf)
                confidence_score = current_max - absolute_difference
                score = confidence_score * 1/i * 1/j
                confidence_sum += score
    return confidence_sum / max_confidence_sum
```

5.2.12 Semantic types

The semantic types have the same purpose of the JDIs, but they have different ways to extract information from the texts. Like the JDIs, they also are available in the "tc2011" library and when extracted, they also have the same format of the JDIs.

The way that I extract and compare the semantic types is identical to the way I extract and compare the journal descriptors, the only thing that changes is the way the text is processed.

The functions and methods described above for the journal descriptors are also used to extract and compare the semantic types.

5.2.13 Doc2vec

This feature is also a feature that uses the texts of the clinical trials and of the articles to have a bigger picture of the content.

5.2.13.1 What is a Doc2vec?

The goal of doc2vec is to create a numeric representation of a document, regardless of its length [7]. But unlike words, documents do not come in logical structures such as words, so the another method has to be found.

The doc2vec models may be used in the following way: for training, a set of documents is required. A word vector W is generated for each word, and a document vector D is generated for each document. The model also trains weights for a softmax hidden layer. In the inference stage, a new document may be presented, and all weights are fixed to calculate the document vector.

5.2.13.2 Doc2vec extraction

To extract the Doc2vecs, we used the gensim library. I downloaded a model that had been already trained on Wikipedia to do that. The gensim library simply loads this model and then I can use it to extract all the Doc2vecs from the texts.

5.2.13.3 Doc2vec comparison

The result of the Doc2vec is a vector of 300 numbers. Having two vectors, we need to compare them, to do this, we use the cosine similarity.

The result of the comparison of the two vectors will be a cosine. To obtain this, we need to divide the scalar product of the two vectors by the product of the lengths of the vectors.

The formula is this one:

$$\vec{v1} \cdot \vec{v2} / (\|\vec{v1}\| \cdot \|\vec{v2}\|)$$

5.2.14 Oger

OntoGene is a research initiative which aims at pushing the boundaries of text mining for the biomedical literature [8].

This project made available a library called Oger [9].
The purpose of this library is to identify biomedical terms inside of a file. Having these terms, I can then count how many correspondences are there between the texts of the clinical trials and the texts of the articles.

Using a database in order to identify the biomedical terms, the OntoGene entity recognition library can recognise useful words. It also has a mechanism that enables it to have a unique id for each term, I use this unique id in order to compare the entities that it finds.

To be able to use this library, I need to save the texts into files first and then use them, I have created a function that does just this, it extrapolates the text of clinical trials and of articles, saves their content on files located in src/tmp_txt_ar for the articles and in src/tmp_txt_ct for the clinical trials. The file names are the id of the clinical trials and of the articles.

After saving the files, the function is able to use the library to determine which biomedical terms each document contains. Having them, I can count how many of them are shared between the documents and I divide the number by the maximum number of biomedical terms present in one of the files.

Summarised in a formula, it is like this:

$$oger\_similarity = shared\_terms(ct, ar)/max\_terms\_number(ct, ar)$$

5.3 Classifiers

I have used two classifiers: a random forest and a neural network.

The dataframe on which I train the classifier is composed by the features described above. I have split it into training set and testing set, the first one contains 2/3 of the rows and the second contains 1/3 of them.

Each row is composed by all the features for a pair of article-clinical trial.

5.3.1 Random Forest Classifier

The random forest is a powerful machine learning algorithm that has a decision tree as the basic building block [10].

The random forest classifier that I have used is the version made available by the scikit-learn library on python.

5.3.1.1 Decision Tree

A decision tree is a tree-model non-parametric supervised learning method used for classification and regression [11].

Each internal node of the tree represents a test for an attribute and the branches (outputs of the node) are the outcomes of that test.

The leaves of the tree represent the final result. In the random forest classifier, as an example, the results are the classes that have been identified, in our case we have only 2 classes, 0 or 1 (the authors are not the same person or the authors are the same person).

The decision tree then, like the name suggests, decides in which class a particular elements needs to be in, and it does this by testing the attributes of that particular element.

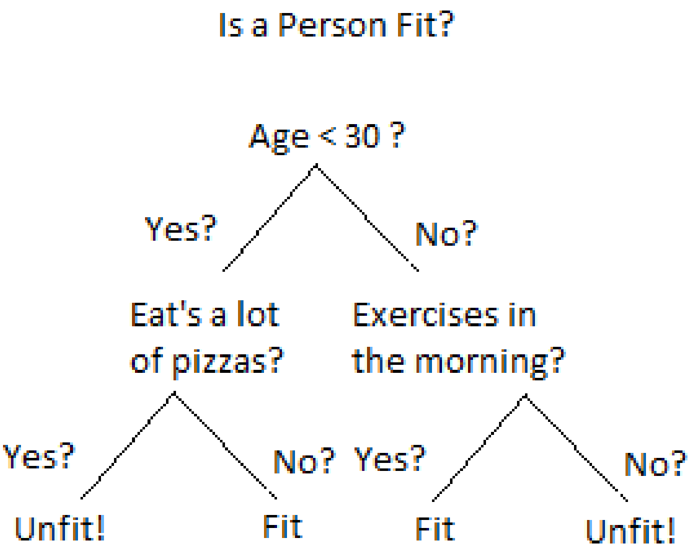


Figure 5.1: Simple Decision Tree

5.3.2 Neural Network

A neural network is a machine learning algorithm whose functioning is inspired by the mechanisms of the neurons. [12].

The objective of it is to perform cognitive functions our brains can perform, like problem-solving.

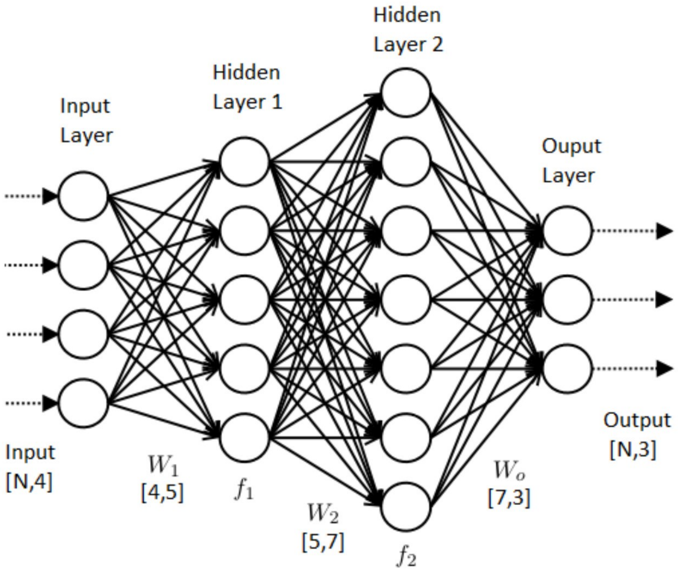


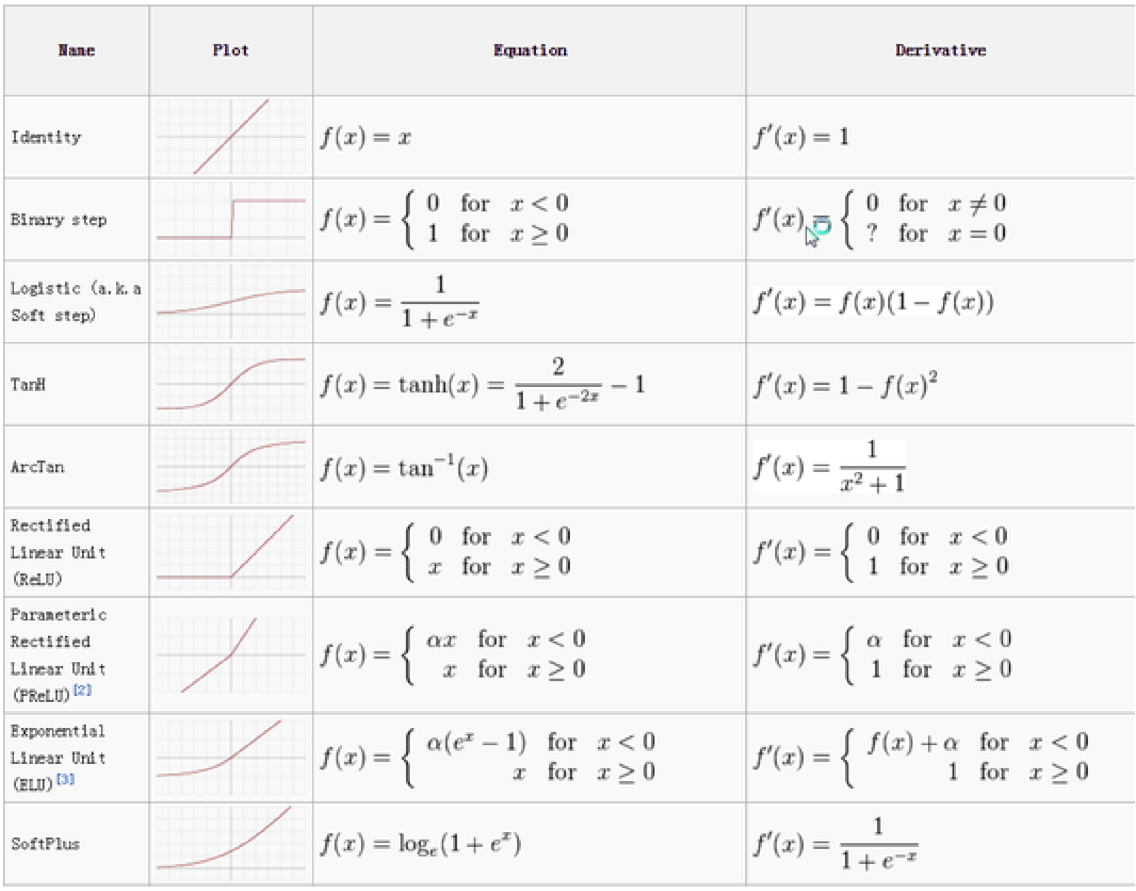
Figure 5.2: Example of a neural network

The components of a neural network are neurons. These neurons make up a network that consists of connections, each connection transfers the output of one of the neurons to the input of another one of them.

Each connection has an associated weight. It represents the strength of the connection. The weight of all the inputs is summed and then an optional bias is added to get the resulting value in a certain node:

$$result = \sum(weight * input) + bias$$

The result obtained then determines whether a node has been activated or not, in order to determine that, an activation function is used.



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 5.3: Activation Functions

The network of a neural network is composed by inputs, hidden layers, and outputs.

The input layer is the first layer. It takes the values, and it passes them to the next layers. The values passed do not have to be transformed here.
In our case, the input layer is composed by all the features listed in the section above.

The hidden layers are layers of neurons placed between the input and the output layers, they apply different transformations to the data they receive as input.
All the neurons in a hidden layer are connected to each and every neuron in the next hidden layer, we then have fully connected layers.
Hidden layers are only necessary if the attributes are non-linearly correlated [13].

The output layer, like the input layer, can only be one for the network. The output layer produces the result we need.
In our case, the output layer is composed by two neurons, one that represents that authors are not the same person, and the other one that represents that the authors are the same person.

5.3.2.1 My Neural Network

I will be using a multilayer perceptron neural network (MLP).
An MLP neural network is a feed-forward neural network [14], the purpose of a feed-forward neural network is to approximate a function, in my case, the function's goal is to assess whether or not the two authors are the same person.

Feed-forward neural networks are the most used neural networks and one of the simplest, their peculiarity is the fact that the connections between the nodes do not form a cycle. The connections only go forward, they only move in one direction. When the connections move also in other directions and form circles, you have a recurrent neural network.

An MLP has at least one hidden layer, the neurons in this neural network use non-linear activation functions.

The final architecture of my MLP neural network is:

- 1 input layer consisting of 14 neurons
- 1 hidden layer consisting of 8 neurons
- 1 output layer consisting of 2 neurons

5.3.3 Support-vector machine

The support-vector machines are supervised learning models with associated learning algorithms.
These learning models aim to find a plane that best separate positive and negative points.

So, this machine learning technique aims to set decision boundaries so that they can divide the points in classes in this way. To do so, it uses support vectors, data points that lies closest to the decision surface.

SVM can be linear (LSVM) or non-linear. LSVM can be used when classes are linearly separable, in the other cases, non-linear methods will be far more effective.

Kernel functions are used in order to transform the data and separate the classes effectively. Some notable mentions are the sigmoid kernel and the polynomial kernel.

Chapter 6

Results

6.1 Journal Descriptors

As said previously, there are many ways to compare the journal descriptors and the semantic types.

The results I get for the various comparison methods are shown in the Figure 6.1.

Those results have been obtained removing the sts_similarity, on the x axis, we have the number of journal descriptors used for the comparison and on the y axis, we have the average precision obtained.

The results of the classifiers have been obtained with 4 journal descriptors using the confidence mode of comparison.

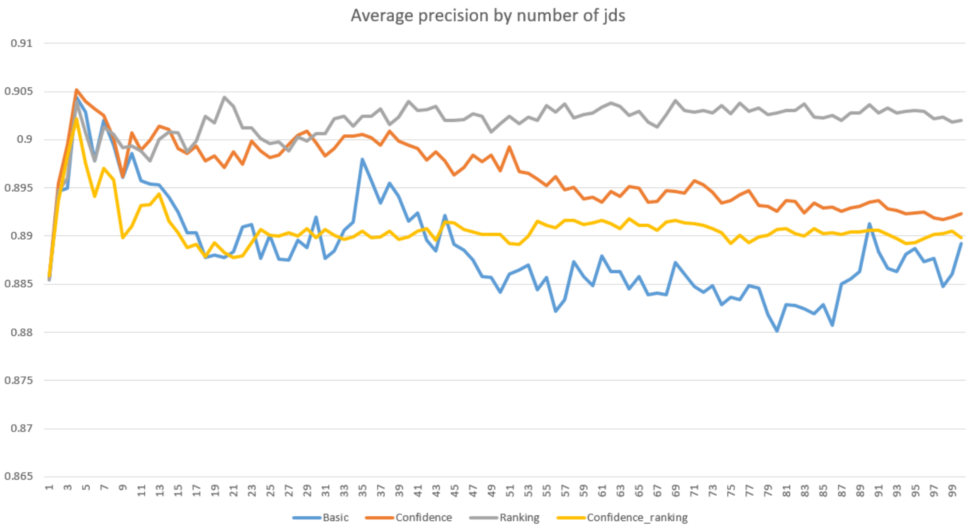


Figure 6.1: Average precision by number of journal descriptors

6.2 Semantic Types

As for the journal descriptors, the same goes for the semantic types.
The results I get for the various comparison methods are shown in the Figure 6.2.

Those results have been obtained removing the jds_similarity, on the x axis, we have the number of semantic types used for the comparison and on the y axis, we have the average precision obtained.

For the results listed below, I used 1 semantic type (only the one with the most confidence) and confidence_ranking as the method of comparing them.

From the graph we can see that, unlike for the journal descriptors, for the semantic types, we are not choosing the best result, that is because semantic types and journal descriptors are both text categorisation tools and it happens that using both features is not yielding the best result.

It can possibly be because they carry the same information and it does not help the classifier, in the case of using the best of the jds and the best of the sts, we obtain a result that is only slightly different from the result obtained using the best number of jds.

Using 4 journal descriptors comparing them with confidence mode and using 1 semantic type and confidence_ranking as comparison mode, I obtain slightly more than 91% of accuracy with the random forest classifier.

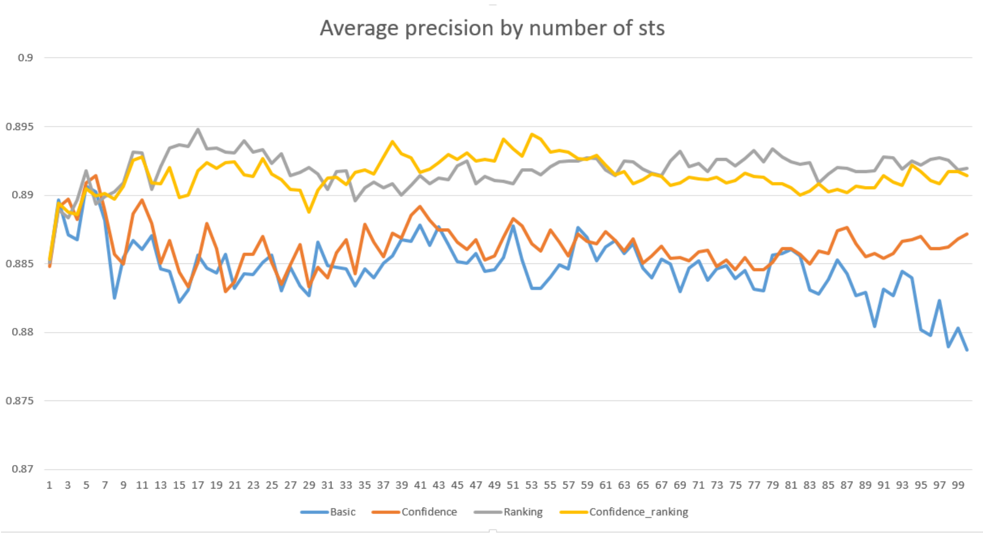


Figure 6.2: Average precision by number of semantic types

6.3 Features' Importance

Here I have two tables that show the importance of each and every feature used, ordered by the importance of the feature.
The average merit of the jds and sts changed heavily when changing their numbers and their comparison mode.

This table shows the results using 4 jds confidence and 1 sts ranking, the best combination I have managed to achieve:

Rank	Feature Name	Average merit
1	Journal Descriptors	0.2197245572233773
2	Full First Name	0.1766520709483555
3	Ambiguity Score	0.1762555733773142
4	Doc2Vec	0.1351088510903348
5	Year Difference	0.0739235541334557
6	Last Name Length	0.0587440960341742
7	Organization	0.0408775957164375
8	Initials	0.0387203592971959
9	Oger	0.0315647356333385
10	City	0.0150215936265885
11	Country	0.0144914564307615
12	Semantic Types	0.0129033769942020
13	Type of Organization	0.0058563475881063
14	Email	0.0001558319063575

Table 1: Table of Features' Importance with 4 jds confidence and 1 sts ranking

We can see that the most valuable features are the one that exploit the text in order to find similarities between the two texts.(semantic types seem to be not important, but we will see in the next table that this is not always the case).

Other important features are the Full First Name and the Ambiguity Score.

In the following table, I have chosen to show what importance these features have when we use the maximum yields of jds and sts, I take the values from the graphs above.
In this table we will be using 4 jds confidence and 17 sts ranking:

Rank	Feature Name	Average merit
1	Journal Descriptors	0.1858369752041338
2	Semantic Types	0.1607814165004505
3	Full First Name	0.1598160502305225
4	Ambiguity Score	0.1515304762461282
5	Doc2Vec	0.1056268755867230
6	Year Difference	0.0617868577590504
7	Last Name Length	0.0535248516230384
8	Organization	0.0360699911713859
9	Initials	0.0316056469670024
10	Oger	0.0231350870016193
11	City	0.0126506679815695
12	Country	0.0125512950394997
13	Type of Organization	0.0049872969943188
14	Email	0.0000965116945569

Table 2: Table of Features' Importance with 4 jds confidence and 17 sts ranking

As you can see, now the semantic types have almost the same importance as the journal descriptors, changing the jds/sts numbers and modes, heavily affect these features importance.

Unfortunately, in this way, I get 0.905965 as accuracy, meanwhile with only 1 sts ranking I get 0.910278.

6.4 Result of the Classifiers

In this section I present the results I have obtained using a random forest classifier, a neural network and an svm classifier. Each one using the average of 50 10-fold cross-validations. The results have been obtained using 4 jds confidence and 1 sts ranking.
The new features used in this project are the Doc2Vec and oger similarity.

6.4.1 Random forest results

Here we have the results I managed to obtain using a standard random forest classifier:

	baseline	+Doc2Vec	+Oger	+Doc2Vec +Oger
Precision	0.900173	0.907276	0.902175	**0.910278**
Recall	0.899900	0.907047	0.901867	**0.909997**
F Score	0.898862	0.905860	0.900764	**0.908949**

Table 3: Random Forest Classifier results

6.4.2 Neural network results

Here we have the results I managed to obtain using a neural network:

	baseline	+Doc2Vec	+Oger	+Doc2Vec +Oger
Precision	0.884551	0.886346	0.886105	**0.886610**
Recall	0.885467	0.887294	0.886960	**0.887626**
F Score	0.882534	0.883844	0.884071	**0.884391**

Table 4: Neural Network results

6.4.3 Support-vector machine results

Here we have the results I managed to obtain using a support-vector machine classifier:

	baseline	+Doc2Vec	+Oger	+Doc2Vec +Oger
Precision	0.881381	**0.888117**	0.881863	0.888107
Recall	0.881640	**0.888805**	0.882202	0.888769
F Score	0.879843	**0.886520**	0.880347	0.886491

Table 5: SVM classifier results

6.4.4 Classifiers comparison

Here we have the results of the classifiers, we can compare them here easily:

	random forest	MLP neural network	support-vector machine
Precision	**0.910278**	0.886610	0.888117
Recall	**0.909997**	0.887626	0.888805
F Score	**0.908949**	0.884391	0.886520

Table 6: Classifiers comparison

Chapter 7

Installation

In this chapter I will explain how to use my scripts. The repository is here :
https://github.com/MatteoBre/AND_clinical_trials.

In order to be able to use this script, you have to download all the clinical trials (they weigh too much to be made avalable on Github). You can download them here:
https://clinicaltrials.gov/AllPublicXML.zip.

After downloading this .zip archive, you have to extract its content in the folder called "AllPublicXmL". You should then have an hierarchy like "AllPublicXML -> NTC0000xxx -> NCT00000102.xml". Copy the "AllPublicXML" folder into the "src" folder.

You will need to download the text categorization library, here's the link
https://lexsrv2.nlm.nih.gov/LexSysGroup/Projects/tc/2011/release/tc2011.tgz. Extract it and place the "tc2011/data" folder in "src/java_libraries".

To be able to use the Doc2Vec feature, you will have to download the pre-trained model, here's the link: https://ibm.ent.box.com/s/3f160t4xpuya9an935k84ig465gvymm2. Extract the content to "src/gensim", you then have "src/gensim/enwiki_dbow/doc2vec.bin" and the 2 linked files.

Another thing to do is to write the path to your java (usually C:\Program Files\Java\jdk-11.0.1\bin\java.exe for Windows or /usr/bin/java for linux/mac) in the file "java_path.txt".

Chapter 8

Discussion and Conclusion

This project aims to solve the author name disambiguation problem in pairs composed by articles and clinical trials. Unlike what has been done precedently, we are comparing different types of data.

The methods of extraction for the information are different, and in some cases there cannot be comparison because fields that are present in the MEDLINE articles are not present in the clinical trials or vice versa.

In spite of this, the precision of the classification is not very different from the comparison of article-article. That means that, even though the data are formatted in a different way and sometimes data cannot be retrieved, we still obtain roughly 90% success in assessing whether or not the authors in the clinical trials and in the articles are the same.

In this project new attributes have also been introduced.
The Doc2Vec attribute has been helpful in order to achieve the final results. I introduced it because the attributes that extracted information from the text (and title) of the articles and clinical trials have been very successful in predicting the final result.
The Doc2Vec also extracts information from the texts, and it has been one of our best predictors.

The Oger attributes tries to identify keywords in the texts (and titles), comparing the resulting keywords it searches for matches. It has been useful, but it is not as successful as I thought. It only slightly improves the classification.

The missing or incomplete information has made more difficult the process of comparing article authors and clinical trial authors, but the attributes that extract information from the text have not been severely impacted.

I believe that author name disambiguation algorithms can be done using different sources of data, in my case, using article-clinical trial pairs instead of article-article pairs has not had severe effects on the precision of the algorithm.

Bibliography

[1] Rezarta Islamaj Dogan, G. Craig Murray, Aurélie Névéol, and Zhiyong Lu. Understanding pubmed user search behavior through log analysis. *The Journal of Biological Databases and Curation*, 2009.

[2] Min Song, Erin Hea-Jin Kim, and Ha Jin Kim. Exploring author name disambiguation on pubmed-scale. *Journal of Informetrics*, 2015.

[3] Jie Tang, Alvis C.M. Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[4] In-Su Kang, Pyung Kim, Seungwoo Lee, Hanmin Jung, and Beom-Jong Youb. Construction of a large-scale test set for author disambiguation. *Information Processing Management*, 2011.

[5] Dina Vishnyakova, Raul Rodriguez-Esteban, Khan Ozol, and Fabio Rinaldi. Author name disambiguation in medline based on journal descriptors and semantic types. *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining*, 2016.

[6] Dina Vishnyakova, Raul Rodriguez-Esteban, and Fabio Rinaldi. A new approach and gold standard toward author name disambiguation in medline. *Journal of the American Medical Informatics Association*, 2019.

[7] Gidi Shperber. A gentle introduction to doc2vec. https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e, 2017.

[8] OntoGene website. http://www.ontogene.org.

[9] Oger repository. https://github.com/OntoGene/OGER.

[10] Will Koehrsen. Random forest simple explanation. https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d, 2017.

[11] Chirag Sehra. Decision trees explained easily. https://medium.com/@chiragsehra42/decision-trees-explained-easily-28f23241248, 2018.

[12] Armaan Merchant. Neural networks explained. https://medium.com/datadriveninvestor/neural-networks-explained-6e21c70d7818, 2018.

[13] Ahmed Gad. Beginners ask “how many hidden layers/neurons to use in artificial neural networks?”. https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks- 2018.

[14] Nitin Kumar Kain. Understanding of multilayer perceptron (mlp). https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f, 2018.