POLITECNICO DI MILANO

School of biomechanics and biomaterials engineering

E - HEALTH PROJECT: SPECIAL PART

FINDING AND DEVELOPMENT A CATEGORIZATION METHOD FOR AN APPLICATION DATABASE: MYHEALTHAPP.COM

Course of E-Health Methods and Applications

Prof. **Alessia Paglialonga**
Prof. **Simona Ferrante**
Prof. **Enrico Caiani**

Saverio D'Amico - 875160
Matteo Dell'Era - 875427

SUMMARY

1. INTRODUCTION 3

2. MATERIAL & METHODS 3

Creation of Training&Test Set 3

Extract data from myhealthapps.com website 3

Create new MeSH terms list 3

Prepare terms list for batch MetaMap *4*

Select MetaMap parameters and save output *4*

Create new MeSH terms list *4*

Processing app description list with MetaMap 4

Formatting App descriptions for batch MetaMap processing *4*

MetaMap parameters *5*

Processing output 5

Analyze XML output *5*

Filtering of the score *5*

Discrimination of not-medical application *6*

3. RESULTS AND STATISTICAL ANALYSIS 6

Statistical analysis 6

Results of other algorithm *8*

Problems 8

4. CONCLUSIONS 9

Future development 9

1. INTRODUCTION

The number of healthcare Apps in the Google Play Store and Apple store is increasing. For this reason, it is important find automatic methods to regularize and categorize them to allow for better patient information. In this part of the project the aim is to characterize all the Apps with an English description of MyHealthApps website. The selection of Apps with an English description is made by filter option present in the website.

2. MATERIAL & METHODS

The Apps are taken from myhealthapps.net, this website take the apps from the Google Play Store and the Apple Store recommended by healthcare communities from all over the world. The code is developed in Python using Jupyter Notebook, a web application that allows you to write and compile live the code, both of them included in the Anaconda Bundle Distribution. For characterization of the Apps MetaMap is used, a highly configurable program developed at the National Library of Medicine (NLM) to map biomedical text to the UMLS Metathesaurus or, equivalently, to discover Metathesaurus concepts referred to in text. In our case Batch MetaMap is used for process apps description.

Creation of Training&Test Set

Before starting to code, 150 apps are selected from all the English database, to create our test database. After reading through all app descriptions, a manual classification is done on all the 150 apps using 15 Medical Specialties plus the "across species" classification, for apps with multiple category medical content. Our test set helped us to see the quality of our script for apps classification by comparing our automated output with the manual.

Extract data from myhealthapps.com website

The Name of the app (**appName**), the description (**appDesc**) and the summary(**appSum**), are taken from the html source code of each apps in myhealthapps.com English database with an automatic script, ***extract_attributes***. This script encodes all the string from Unicode to ASCII and remove all newline characters ("\n") for have all the parameters in one single line and write all the extracted information in an Excel file. Summary is taken because it contains a lot of information that can improve the number of match with the MeSH terms list given by professors. The function ***create_listURL*** was used to obtain the html address of all the applications.

Create new MeSH terms list

As seen in previous project, the number of match between MeSH terms list and MetaMap output of apps description increase a lot if also MeSH terms list is processed with MetaMap. The steps, already descripted in the report of the first part, required for process MeSH terms list with MetaMap are the following.

Prepare terms list for batch MetaMap

Initially the script assembles all terms files into a single text file in accordance with the MetaMap standard. This included reading all the terms files, putting them in order in a single file, eliminate all non-ASCII characters as well as adding the terms category ID in front of each line. For do all of this step the script is: ***terms_to_metamap***.

Select MetaMap parameters and save output

Some MetaMap's parameters are selected to create an output that would be easy to process:

Single Line Delimited Input w/ ID	This parameter is used to combine many different sources into a single file using different ID. This ID is then also present in the output file which enables a better keep track of multiple sources in a single batch process.
Formatted XML Output (--XMLf)	With this parameter MetaMap create an output formatted in XML

Create new MeSH terms list

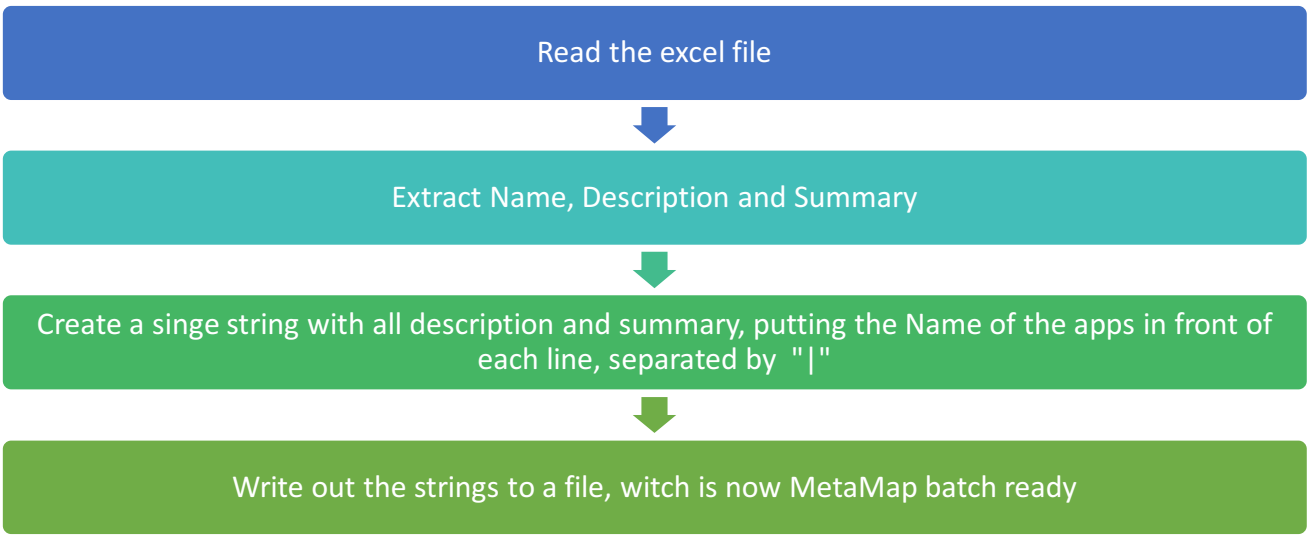
The script ***create_new_terms*** automatically subdivide the XML output in each MedicalSpecialties present in MeSH terms list given by professors. For increase the number of matching the original MeSH terms list and the new one is merged with the script ***create_merged_terms***.

Processing app description list with MetaMap

The goal of the project was to classify our Apps database with the given list of Medical Specialties. App descriptions, name and summary are used for creating the input file for batch MetaMap. The first categorization was on Test database to check the specificity of the algorithm; then all database is processed with batch MetaMap.

Formatting App descriptions for batch MetaMap processing

All app descriptions were given in the form of an excel sheet. To process them and format for MetaMap processing the script works as follow:



For improve the number of match between MetaMap App output and Metamap MeSH terms list output also the name of the apps is included in the description, because some Apps have a very short description and the name could also match with a MeSH term and it can help the classification of the app. Script used: ***excel_to_metamap***.

MetaMap parameters

For the apps description file, we put the same MetaMap parameters used for MeSH terms list described in the previous paragraph, so we obtain an XML formatted out.

Processing output

With new MeSH terms list and apps description output, the categorization of our database using the 15 Medical Specialties proposed by professors can start. The script using for processing output is: ***compute_categorize***.

Analyze XML output

The XML formatted output is a large file that contains various information. *CandidatePreferred* and *Score* was extracted and used for looking at correlations and patterns between MeSH terms list output and Apps output. For avoid the acronyms problems was added a method to ignore words of less than 4 letters if they are all capital letters.

Filtering of the score

To best categorize the healthcare apps based on their description and summary alone, was necessary an algorithm for ranking which category is most likely to be correct. The kernel of categorization is based on the following algorithm:

$$STF = \frac{TS}{n} * Q$$

TS is the total score for each category, **n** is the number of match and **Q** is quality parameter of the category matched.

$$TS = \sum score \qquad\qquad Q = \frac{TS}{\sum count * 1000}$$

Here is reported an example of how the algorithm calculated the parameters:

```
Nutrition.txt
{'Food': {'count': 1, 'score': 1000}}
Cardiology.txt
{'Blood pressure finding': {'count': 1, 'score': 1000}}
{'Systemic arterial pressure': {'count': 1, 'score': 1000}}
{'Blood Pressure': {'count': 1, 'score': 1000}}
DiabetesCare.txt
{'Diabetes Mellitus': {'count': 7, 'score': 4455}}
Endocrinology.txt
{'Diabetes Mellitus': {'count': 7, 'score': 4455}}
```

CATEGORIES MATCHED	TS	N	Q	STF
Nutrition	1000	445	1	2.24
Cardiology	3000	1983	1	1.51
DiabetesCare	4455	88	0.63	31.9
Endocrinology	4455	534	0.63	5.25

So, for each application, at all the category find by the script is associated a score, STF. The two categories with the highest STF value are selected as the First category and the Second category of the application.

Discrimination of not-medical application

Two steps were used to discriminate the non-medical application in the test file, schematized in the following table.

Step 1	**No match** in the Mesh terms list Merged
Step 2	**STF > 4** & **Q > 60%** to access the categorization. If no matched category passes, the application is considered not-medical

3. RESULTS AND STATISTICAL ANALYSIS

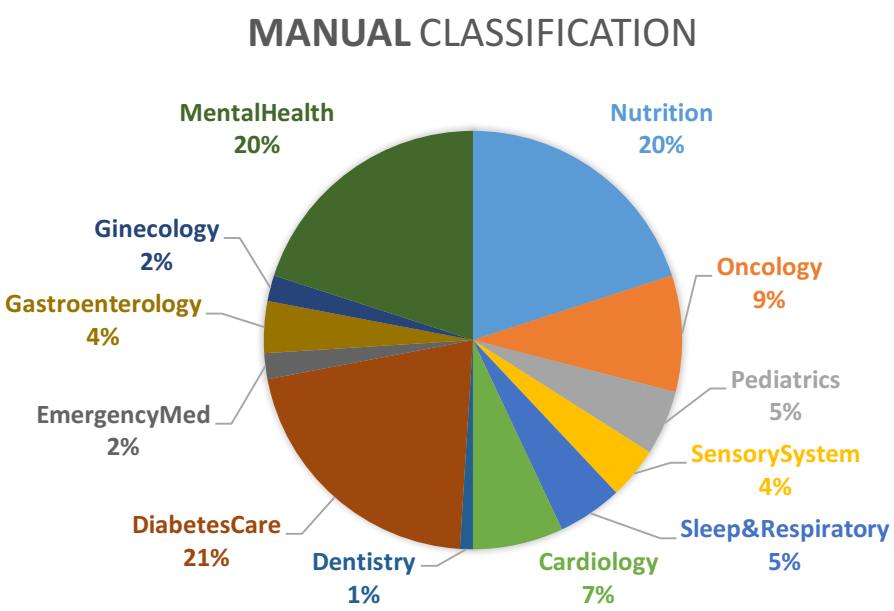
Statistical analysis

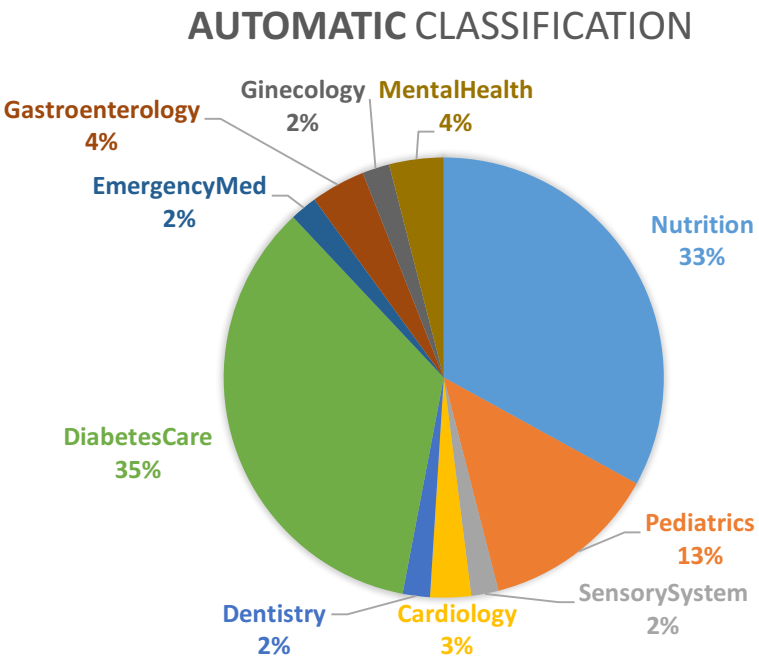
For understand how well the algorithm classify the Test Database, some statistical parameters were calculated:

% CORRECT	correct match/ total number
% NOT-MEDICAL	number of correct not-medical match/ total number of not- medical in manual classification
% MEDICAL	number of correct medical match, included the match with the correct category in second position / number of medical apps in database

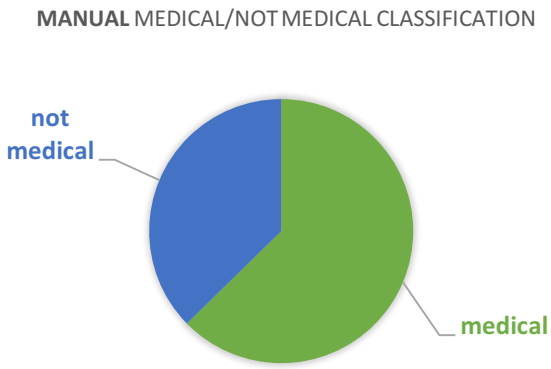
Also a binary classification test was done for analyze the specificity and sensitivity of the algorithm for classify an app, with 1 were consider the medical apps and 0 the not-medical apps. For this test 'condition' represent the manual classification of the test database and 'Test' represent the automatic classification of the algorithm.

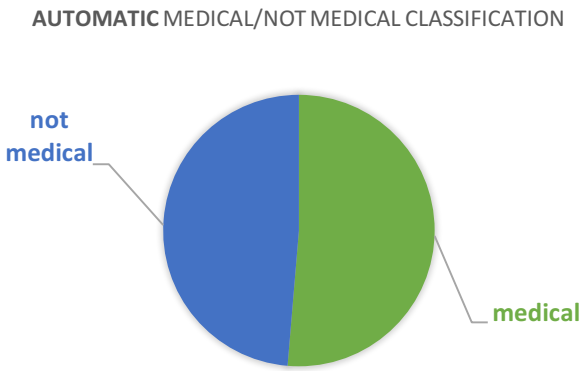
In the following graph are represented the percentage of each Medical Specialties present in the TT Database for manual classification and automatic classification.





This 2 following graph represent the % of not-medical and medical app in manual classification and in automatic one.





The following table contains the result of the comparison between manual classification and automatic one made on Test Database:

FINAL ALGORITHM	% VALUE
%correct	**66.0 %**
%not medical	**80.4 %**
%medical	**55.1 %**

This table shows that the algorithm categorizes efficiently the not-medical apps, but has some problems with medical apps. This lead to a total percentage of correct match of 66%. Here are reported the binary statistical analysis performed on the test database:

TP	FP		
54	9	SENSITIVITY	**65.9 %**
28	45	SPECIFICITY	**83.3 %**
FN	TN		

This algorithm has a high specificity so rarely a not-medical app is categorized as medical; sensitivity is not so high so the algorithm sometimes categorizes as not-medical too many medical app.

Results of others algorithm

The final algorithm was selected between another possible algorithm. The decision of witch algorithm select was made by consider the algorithm with the highest and equilibrated % of correct match.

	1	2	3	4	5
PARAMETERS	Q, TS	Only TS	TS, Q	TS, Q, n (filter 5)	TS, Q, n (filter 5)
ALGORITHM	$STF = Q$	$STF = TS$	$STF = TS * Q$	$STF = \frac{TS}{n} * Q$	$STF = \frac{TS}{n} * Q$
RESULTS					
% tot	61.22 %	64.63 %	64.63 %	67.34 %	66 %
% not medical	53.57 %	50 %	53.57 %	89.29 %	80.36 %
% medical	65.31 %	70.41 %	72.45 %	50 %	55.10 %

Problems

One problem is that in some Medical Specialties, like sleep and respiratory care or emergency medicine, there are very small number of terms. For this reason, it is difficult for the code find some match in this category, so it is a good idea to work on some Medical Specialties to improve the number of terms and increase the probability of matching. Another problem is the recognition of not medical apps: the algorithm finds too many not-medical apps, that in manual classification are classified as Medical app, and for improve the Medical/not-Medical classification a large new Medical Specialties, that contain generic medical terms can be created for increase the % correct.

4. CONCLUSIONS

In this project was developed a script in Python that categorize our apps database using a new MeSH terms list of 15 Medical Specialties, created by merging the professor MeSH terms list and the MeSH terms list processed with batch MetaMap. The categorization isn't 100% correct and the problems, described in the previous paragraph, can be solved with the introduction of a new category, Generic Medical Terms, and by adding more terms to some Medical Specialties.

Future development

It is possible to improve the algorithm more and more, finding more and more effective parameters in order to solve some cases of particular applications, for example in the case of descriptions too short or ambiguous. Here are two beta versions of new algorithms designed to solve these problems.

	6	7
PARAMETERS	TS, n	TS, n, (Q used only as filter)
ALGORITHM	$STF = \frac{TS}{n}$	$STF = \frac{TS}{n}$
RESULTS		
% tot	66 %	64 %
% not medical	76.79 %	80.36 %
% medical	59.57 %	54.26 %