EXECUTIVE SUMMARY

Nowadays, a lot of companies gather content crawled from several social networks in order to carry out analysis about products, phenomena or even the social life of the users. SecondSync is both a company and a platform that reclaims large amounts of data (crawled from Twitter) in order to produce analytics about TV broadcasts. Twitter has been selected as the single data source to collect such information. The main reason for this decision is that twitter messages are more personal therefore, by taking advantage of this fact, it is possible to extract information such as viewers' opinions about specific TV-programmes, popular actors, popular programmes and TV trends.

Nevertheless, in order for an analysis to be successful, the SecondSync platform must use data which is strictly related to TV broadcasts. However, this is not always the case because micro-blogging messages could be relevant to any topic. This is a major observation and a big challenge that this project must deal with by applying specific scientific methods to big tweet datasets in order to identify TV-related activity accurately. The existing methods that SecondSync platform uses, can already track such information but they produce many false positives, which affects the accuracy of the analytics.

This project explores filtering techniques in order to analyse Twitter content and extract accurate information that is exclusively relevant to TV-broadcasts. As had been mentioned above, existing methods tend to lead to results that consists of many false positives and because of that, such datasets may be irrelevant. The research made for this topic unveiled the fact that there are more specialized practices and methods for extracting TV-series/Movies information from Twitter. Nevertheless, these techniques could be improved further. Another major goal of this project is to determine and develop the necessary techniques to improve existing filtering methods. In addition, the developed system should be less complex in order to cope with a real-time analysis. Specifically, the developed techniques should be both accurate and efficient. That was accomplished in two distinct phases.

Firstly, taking advantage of TV subtitles, keywords, that describe accurately a certain TV-series/Movie, are extracted. Such terms are used in order to query Twitter for content that may be relevant to TV-series/Movies. The returned results contain many false positives, which are reduced by passing them through a filtration procedure. This filtering system is the second phase of our project where a machine learning model is responsible for reducing the false positives. For both of the above phases, major research was carried out on how to achieve the desired results. The improved system was constructed programmatically, and the evaluation that had been made, validates the fact that it is both accurate and efficient.

The main contributions that this projects offer are:

- The development of an algorithm that produces keywords from TV-subtitles. The produced keywords describe the corresponding TV-series accurately.
- The design and development of a filtering mechanism that identifies tweet messages relevant to TV-series/Movies. This system is an improvement of the research presented in [6]. This system solves the problem of the false positives. Furthermore it is less complex, thus it could be used in a system that produces real-time TV analytics.

Table of Contents

ABBREVIATIONS LIST

ORM:	Object-Relational Mapping
NLP:	Natural Language Processing
POS:	Part-Of-Speech
NER:	Named Entity Recognition/Recognizer
TF:	Term Frequency
IDF:	Inverted Document Frequency
ML:	Machine Learning
TS:	Training Set
TV:	Television
IMDB:	Internet Movies Database
OCR:	Optical Character Recognition
SCV:	Comma Separated Values

1. Introduction

Nowadays, social networks constitute the best source for mining either general information or personal opinions. Social networks are online communities where organisations, people and other groups of individuals live all together and publish content. From the opinion mining point point of view, the daily published content can be used in order to extract conclusions, opinions or facts for various issues such as earthquakes[1], traffic[2], epidemics[3], politics[4], news[5] or TV series[6].

According to [7] a lot of active Internet users have abandoned the traditional means of communication such as forums, blogs and mail lists and turned to use more modern platforms to publish their personal opinions and express themselves. These platforms consist of online social networks or micro-blogging social networks such as Twitter. In Twitter, users are able to publish posts consisting of no more than 140 characters. Those posts are widely known as Tweets and contain opinions, web links, news and facts. Twitter provides developers with an online api which could be used to retrieve large corpus of tweets. As long as this information is easily accessible from everyone, the analysis of these datasets is possible and indeed it can be made. As a result, valuable conclusions and opinions can be extracted from the growing audience of Twitter and be used either for marketing or social purposes.

SecondSync Ltd. is a company interested in analysing Twitter content related to TV broadcasts. The company owns a platform which creates linkages between micro-blogging messages and UK airings such as TV-series, Talk-shows, Movies, and Documentaries etc. In more detail, the main goals of the system are:

- Efficiently find the most popular broadcasts based on Twitter traffic during specific periods of time.
- Utilize this information and provide TV analytics.
- Gather and classify Tweets related to certain broadcasts.
- Use these tweets and apply analysis on viewers' sentiments about each broadcast.

These elements are quite important for marketing purposes and extremely useful for channels, advertisers, producers or journalists.

Twitter has been chosen from SecondSync as the basic source for extracting information relevant to TV broadcasts for many reasons, explained below. First of all, this social network has more than 100 million active users worldwide. It is remarkable that currently, 20 million users who publish 6 million of tweets in a daily basis and are located in UK. Apparently, this content is the perfect dataset to be used from SecondSync in order to extract TV-related information and hence to further process it.

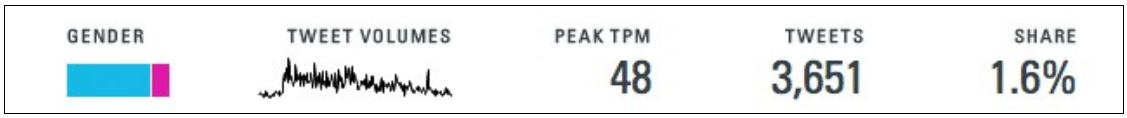
Currently, the company takes advantage of the online web services to capture all the tweets originated from the UK. Subsequently, the developed platform queries the

database containing the micro-blogging messages in order to find tweets relevant to UK broadcasts. The querying is made by using name keywords (extracted from TV subtitles) and the title of the broadcast as search terms. In addition, the platform tries to identify tweets relevant to specific broadcasts aired at a specific moment. The major problem that the existing system address is that it cannot efficiently identify such tweets and returns several false positives.

In order to successfully identify such tweets, technologies such as tag extraction (by using the subtitles of each broadcast) and tweets classification are necessary. The first technology is necessary because it can capture important terms from broadcast's subtitles and not only names as the existing platform does. The second technology should be used in order to construct a classifier which identifies tweets relevant to TV broadcasts and produces only a small amount of false positives.

It is big challenge to develop an improved identification/classification system that excludes false positives among a lot of irrelevant tweets. In addition, the classification should take into mind that there are various types of broadcasts such as series, talk shows, news etc.. Each one of these have different characteristics that could be used for the identification. A general system for identifying messages for all of these types, do not produce the desired results. This research examines the TV-Series/Movies classification.

However, the existing platform does not take into consideration all these details but uses various online apis that are built for general purpose systems and do not overcome all the difficulties. Using the same technics and the same classifier for all the types of broadcasts is not a good solution. This method leads to too many false positives and they should be reduced. One example of this problem presented graphically below:



Picture 1: This image represents a TV-analysis that contains a lot of false positives.

These statistics are taken from the SecondSync platform and represent the real-time analysis for a specific TV-series broadcast. The graph above represents how many tweets (related to certain TV-series) had been published during the airing time of the broadcast. More specifically, the x-axis represents the time and y-axis represents the number of tweets published. It is important to notice that this graph represents a classification with many false positives. This is the case because the graph has many peaks, one after another. If we relate these peaks to viewers' behaviour, it is obvious that every tiny intervals of time Twitter users were posting something about an event that happened in the current TV-series. Despite the fact that the viewers' tend to publish tweets at the same time during the broadcast (this fact is presented with peaks in the graph) when an event had happened, the peaks did not appear so frequently because these facts were not happening continuously. As a result, the above graph takes into account tweets that they are not relevant to a specific TV-Broadcast. Theoretically, a classification without false positives should be presented by smooth peaks as follows:

Picture 2: This image represents a TV-analysis that does not contain a lot of false positives.

The false positives problem affects further the functionality and the aims of the developed platform. Apart from identifying tweets relevant to TV broadcasts, another goal of SecondSync platform is to process these tweets in order to carry out an effective analysis in viewer's sentiments about certain broadcasts. In addition, by having an inaccurate tweets subset, the platform is not able to carry out an accurate sentiment analysis. As it has been mentioned above, this fact makes even more essential the need to develop efficient classifiers for filtering the different types of broadcasts among huge tweet datasets. Moreover, according to company's director the available subtitles dataset include important information (subtitles are not used enough in the current platform) that can be used not only for searching for tweets that are relevant to TV broadcasts but also for building an efficient identification procedure for the returned results. Furthermore, the research carried out for this topic unveiled the fact that there are more specialized practices and methods for extracting accurate information from Twitter than those methods presently used by the company. Our goal is to develop an efficient classification system that uses provided datasets (from the company) (such as subtitles, television listings, broadcasts information crawled on web) and identify TV-series and Movies broadcasts activity on social networks.
In more detail the aims of this project are:

- Design and Development of efficient Keyword Extraction method that suits subtitles. The results of this phase should be used in order to extract Names or keywords that describe a certain episode of the series or a movie. Consequently, these keywords hwere used as search terms to search the datasets to find tweets that might be relevant to TV-series or movies.

- Research for existing methods that accomplish efficient TV-series/Movies classification. By using this classification a filtering system should be constructed for tracking tweets relevant to TV-series/Movies and separates them from irrelevant ones. This procedure can return many false positives (tweets that are taken as relevant but in reality they are not).

- Determine if the existing methods can be improved and develop new techniques needed to extend the system that is described above. Thus, it will become more efficient, regarding its accuracy.
  There are two distinct phases in which this fact can be accomplished:
  - Take advantage of TV subtitles and the extracted keywords for certain TV series or movies in order to use them to improve the classification of tweets.
  - Design and develop, programmatically, a machine learning model that identifies TV-series and movies tweets and produce reduced false positives. Finally, among other features the classifier should be able to take advantage of TV subtitles in order to increase the accuracy of the system.

- Present a detailed evaluation of the above system and how it can further extended in the future in order to identify tweets that are related to other types of broadcasts such as news.

- Because of the architecture of the existing system, the above goals should not only produce an accurate system, but also efficient in the manner of time consumed to accomplish the whole classification.

In order to explore all these problems in depth and produce some results SecondSync Ltd. provided this research with three distinct sets of information as mentioned below:

- Tweets dataset
- Subtitles dataset
- TV listings dataset for UK TV-channels

So far, the major problems that this research aims to solve had been presented.

This report consists of five chapters. In the next chapter, the basic notions of this research will be explained. Additionally, in the second chapter the basic tools that had been used for the development of this research and the architecture of our system will be presented. In the third chapter it is described our keyword extraction mechanism by using TV subtitles. This phase will produce keywords that are going to be used for searching tweets that might be relevant to TV broadcasts. Finally, in the fourth chapter the filtering mechanism and how our system reduces the large amount of false positives will be explained.

2. Background

The current chapter is describing the general background that the reader requires in order to become familiar with the basic concepts of this research and to understand better the nature of this project. In more detail, this chapter is providing information about the tools and datasets which were necessary for the design and the development of the system that this report examines.

It is clear that the datasets are important elements in order to build improved mechanisms and solve the false-positives problem. On the other hand, some tools are important as well as they will improve the efficiency of our system in a manner of time. The time factor is critical because of the fact that the SecondSync platform analyses information in real time. The datasets and the tools used in order for this research to be accomplished are described below:

2.1 Datasets

In order to build an efficient and accurate TV-series/Movies classification system, some datasets are necessary. Datasets are the core of the current project and will be used in order to build an accurate classification system. Moreover, many of them will be used in order to build our Tag extraction system that will lead to a better classification system. All the datasets that had been used as well as the main reasons of how these datasets could affect our research in a negative manner are presented below.

2.1.1 Subtitles

Subtitles are textual captions displayed at the bottom side of a TV screen. New-generation television sets have the option of displaying real-time generated subtitles in the native language of the current viewer. This modern technology is adopted after the establishment of the digital TV some years ago and is extremely useful. Such applications are essential because they improve the life of people with hearing difficulties or the life of foreigners. An example of these textual captions can be seen in the following image taken from BBC's web TV.

Picture 3: This picture presents a scene from a news-show that contains subtitles.

As it had been mentioned in the introduction chapter, subtitles represent a very useful dataset for the current project's research. The subtitles generation is feasible by using real-time speech-to-text technologies. These technologies do not provide 100% accuracy hence, some mistakes were noticed during subtitles generation. The most important mistakes that we have noticed during the processing of the subtitles were unknown words consisting of known words (e.g “in just” generated as “injust” or “I” “am” generated as “Iam”). This could be the major error because of the fact that the current platform could recognise “Iam” as a Name word and add it in the Name Entities List for a current broadcast. Such problems have been resolved and are explained in the following chapter.

Last but not least, because of copyrights issues, TV channels have not the intention to share the subtitle datasets in textual form. However, the existing system uses a robust mechanism that takes screenshots of TV scenes. These screenshots are taken every regular intervals of time and are stored as picture files in the system. Subsequently, the underlying system converts the pictures to textual subtitles by feeding the images into an OCR platform. Current OCR platforms offer amazing accuracy and as a result our dataset do not contain mistakes that originated from this system.

2.1.2 Metadata or TV-Listings

Meta-datas as defined in [2] are collections of structural text that contain information for TV-airings. Such information includes show titles, broadcast's start/end time and date, broadcaster, unique show id, actors, characters, producers, directors, genre, publish year, language of the broadcast and several other information relevant to the specific broadcast. Depending on the source of such meta-data which had been retrieved, their format can be slightly different but always relies on a specific format. An example is presented below that presents TV-Listing information in XML format:

```
<channel id=”bbc1” source=”BBC” date=”14/07/2012”>
        <programme>
                <title>EastEnders</title>
                <description>
                        Kat discovers her mystery man.
                </description>
        </programme>
<channel>
```

TV-listings are a subset of meta-data. Furthermore, these contain the timetable schedule of the television or radio for specific intervals of time e.g. daily, weekly or for longer periods of time. Nowadays, channels publish their schedule automatically and as a result TV-listings are available electronically.

The TV-listings dataset, which had been provided for the purposes of this research used the .csv format which is easily manageable from database systems. The only difference from the xml meta-datas is that the tags exist in the beginning of the document and the content is separated by commas. Our TV-Listings dataset contains the following tags:

[TransmissionID] [Title] [Start Time] [End Time]

The major problem with the TV-listings provided by the company is the following: the

dataset contains neither the information about the characters of each broadcast nor the description of each broadcast.

Furthermore, some titles of our dataset denote the title of the episode, but not the proper title of the series. Lack of such information cannot guarantee the construction of an efficient classification system because, as explained later, our classifier is based on such features. As a result, in our system actors, cast, and the description of the episode/movie is crawled from the web or is captured by using web services linked to imdb.com[1].

2.1.3 Tweets

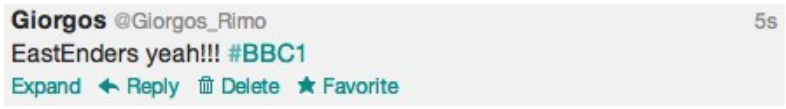
As mentioned in the introduction chapter, the major goal of this project is to identify tweets relevant to TV broadcasts with high accuracy. As a result, the most important dataset that we owe is the 18.000 Tweets stored in our database. Tweet is a small-length message (140 characters) that users of the social network Twitter can publish in their public profile. Despite the small size, these messages usually represent opinions, facts or news. As a result, by analysing huge amounts of Tweets, it is possible to extract what the majority of the users believe for various of topics such as politics, sports or television programmes.

One major problem in the analysis of such tweets, when it is done from a computer, is that the small length of them does not reveal exactly what the current tweet is about. A clear example about this, is the following:

Picture 4: This image presents two controversial tweets.

Classifying such tweets and identifying whether they are relevant to TV broadcasts or not is a big challenge and no perfect accuracy had been achieved so far. However, there are unbiased characteristics for each tweet which could be used in order to achieve a good accuracy. For instance, if the first tweet is published during the airing time of the HOUSE series, then it is likely that this tweet is relevant to the corresponding TV series. (*broadcast time feature*)

Another useful element which adds additional information to each tweet, is that the users are encouraged to use a character called hashTag(#) followed by a keyword describing their post. Apart from this use, hashTagged terms are used in Twitter as search terms, in order to facilitate users when they search for specific topics in Twitter. For example:



Picture 5: A tweet that contains hashtag.

1 imdb.com stands for Internet Movie Database. This site contains all the information needed for our system such as, actors, cast, short description and year.

As you may see the hashTag is very useful adding some information related to each post. It could be used as a core element in our classification procedure in order to identify tweets relevant to TV broadcasts. It should be mentioned that the tweets datasets are captured by a service that SecondSync uses. This service is provided by a company which has the exclusive right to capture all the tweets that are published daily. SecondSync has a paid subscription to this service and as a result this research was provided only with a limited set of tweets due to copyright reasons.

These were some examples of how Twitter works and some of the difficulties that this research may face. Notice that the tweets above are relevant to TV-series but, there are a lot of other broadcast types which have different characteristics from TV series such as broadcasts relevant to sports. In order to identify tweets relevant to sport shows, such as a football match, we cannot use the broadcast time feature presented above. This feature is not valid because, the length of a football match is not predefined. This may be the case for example if a player gets injured, the match would last longer. Furthermore, even if the game ends the Twitter users may continue to talk about it much after the actual airing time of the match.

Unfortunately, there is no efficient way to track tweets relevant to TV broadcasts by using common characteristics or the same classifier. As a result, the different types of TV broadcasts are presented below and their main characteristics are namely defined. Furthermore, it is presented how these characteristics could be used in order to detect tweets relevant to each category.

2.2 Types of broadcasts

-TV series-
A recurring TV broadcast where actors impersonate roles. The characteristics of this type of broadcast could be divided into two separate categories: Static and Dynamic. The most common characteristics of TV series are shown below:

- People who participate in TV series are actors.
- Actors impersonate roles (characters) thus, in each episode the name of the characters does not change but are constant.
- The airing time of the show is predefined.
- Each episode has a predefined plot.

Based on all the above, it is easy to understand that static characteristics are those that do not change and are common among all the episodes such as actors' names, characters names and the airing time. On the other hand, dynamic characteristics are those which are unique for each episode such as the plot. In the current research static characteristics could be crawled from the internet. In addition, dynamic characteristics could be tracked by analysing the subtitles of each episode. In order to track tweets relevant to TV series, these characteristics could be used in order to identify if a tweet contains information relevant to the above characteristics.

-Movies-
These broadcasts are relevant to TV series but they are not recurring. Both of them have a lot of common characteristics such as actors, characters, and predefined airing time. As a result, it is feasible to construct a common system identifying tweets relevant to TV-series as well as movies.

-Sport shows-

These types of broadcasts include all the shows and sport events aired on television. Characteristics of these shows that could be found in tweets relevant to sport shows are:

- Name of teams.
- Name of players.
- Sports terminology.

All the above characteristics are common not only in tweets relevant to sports shows but also in tweets containing opinions or facts about sports in general. The big challenge is to separate those two groups of tweets. This could be done by identifying what are the additional characteristics that tweets relevant to sports shows have. For example:

- The name of the presenter of such shows.
- Start time of the show.
- Channel where it is broadcasted.

These additional characteristics could make the separation feasible. In order to build a system like that, both of the above subgroups of characteristics need to be combined. Firstly, it is needed to take advantage of information existing on web. Such information could contain names, players but also sport terminology such as: football, corner and ball. These datasets should be small and targeted only to shows that will be broadcasted on television in the forthcoming future. For that reason this information should be crawled from sport news sites that contain such information (meaning sports information relevant with matches that are going to be aired on television). By analysing this information (crawled articles and texts) with Natural Language Processing techniques, keywords such as team names or players could be extracted. These keywords are likely to appear in tweets relevant to the specific sport shows or sport matches. Afterwards, these keywords could be used for developing features for a classifier that efficiently identifies tweets relevant to shows. Furthermore, the subtitles displayed on sports broadcasts containing the names of the presenters for each sports game. As a result, this detail could be used to link directly tweets with sports shows.

-News-

This category of broadcasts is completely generic and is more difficult for such information to be tracked in twitter messages, compared to sports shows or movies. The difficulty is based on the fact that news tweets do not have a specific characteristic in which we could rely on and thus, the tracking procedure becomes a very challenging task. Another challenging issue is that news change rapidly in small interval of times, so there is the need to maintain cluster with news happening right now and is possible to be presented in news shows. Furthermore, another problem is that twitter users tend to discuss about the facts presented in news shows after the end of the show. Some of the characteristics that could link news broadcasts to tweets are the following:

- Presenter's name
- Names of the persons invited to the news shows
- Facts happening right now in the world such as politic situations, wars etc.

So far, there is no efficient procedure to detect news tweets relevant to TV broadcasts without a lot of false positives. This is logical because of the nature of news. However, by analysing the subtitles of a news show and crawling information from news website

tweets relevant to news shows could be detected but without much accuracy (many false positives). The analysis of the subtitles with Name Entity recognisers (NERs are software identifying person's names) could map names to presenters in a news show. Finally, developing a classifier identifying tweets including news and names of the show could give a fair set of tweets relevant to television news shows.

-Documentaries-

This category includes shows presenting in detail specific reports about various issues. Such shows have fewer characteristics than other types of broadcasts that could be used in order to identify tweets talking about documentaries. Such characteristics are:

- Airing time
- Meaningful phrases that describe documentary's topic

Airing time could be extracted from TV listings and be used in order to filter tweets published only during the time of the documentary broadcast. Furthermore, Natural Language processing procedures could be used to extract the major subject of the documentary. As previously mentioned, phrases (and not keywords) extracted from the Natural Language Processing procedure could be utilized, in order to find tweets talking about the topic relevant to the current documentary. The big challenge for this category concentrates on the fact that the use of the search terms provided by the NLP mechanism tend to return a lot of messages (tweets) relevant to the current subject but not about the current TV-programme. Notice that combining the above features can likely produce results containing tweets relevant to the documentary but also tweets considered as false positives. The first group could somehow be separated from the other if additional features were used. That separation is discriminating tweets relevant to broadcasts containing for example channel keywords. The above is a simple example about how a classifier can distinguish tweets relevant to television documentaries.

-Talk Shows-

In such shows usually, an interview or a discussion is presented between individuals. The main characteristics of this type of broadcast that could be found in Tweets relevant to television talks shows are:

- Name of persons that participate in the discussion
- Airing time of the show
- Title of the show

The first characteristic could be extracted during the beginning of the show. It is common for the presenter in such shows, during the start time of an interview or a discussion, in order to introduce his guests to the public. Thus, an advantage of this fact could be taken and extract the names of people by applying Name Entities Recognition (see chapter 2.3.2) to the subtitles of the show. Thus, by combining such names and the airing time of the show, tweets relevant to certain talk shows could efficiently be found.

2.3 Tools

This research uses some external tools provided as java libraries in order to achieve the desired goal, to develop an efficient filtering system for tweets relevant to TV-series and movies.

The first phase of our system is to search tweets efficiently which might be relevant to

TV broadcasts. This system is using the provided subtitles and extract keywords used as search terms. These keywords could be nouns or names that had been appeared in the subtitles. In order to identify what part of speech each word is in the subtitle texts, software had been used called Part-of-Speech Tagger. Furthermore, in order to identify named entities such as names of actors or cities that are contained in the subtitles, we use a software called Named Entity recognizer.

As far as the second phase of this research is concerned, a classifier filtering tweets and identifying only the relevant to TV series or movies is needed to be developed. In this phase a software tool called WEKA could be utilized. This is a software package containing machine learning tools such as classifiers, evaluation methods and tools to handle training sets. Last but not least, in order to handle and process the datasets a software tool called Hibernate had been used. All the tools mentioned above are explained in detail below.

2.3.1 Part-Of-Speech Tagging

Part of speech tagging or grammatical tagging is a major procedure in linguistics. It is used in order to identify the part of speech of a certain word appeared in a text. This could be a challenging process because a word could have different grammatical types depending on the context. For instance, the word “talk” could have different part-of-speech in two different sentences:

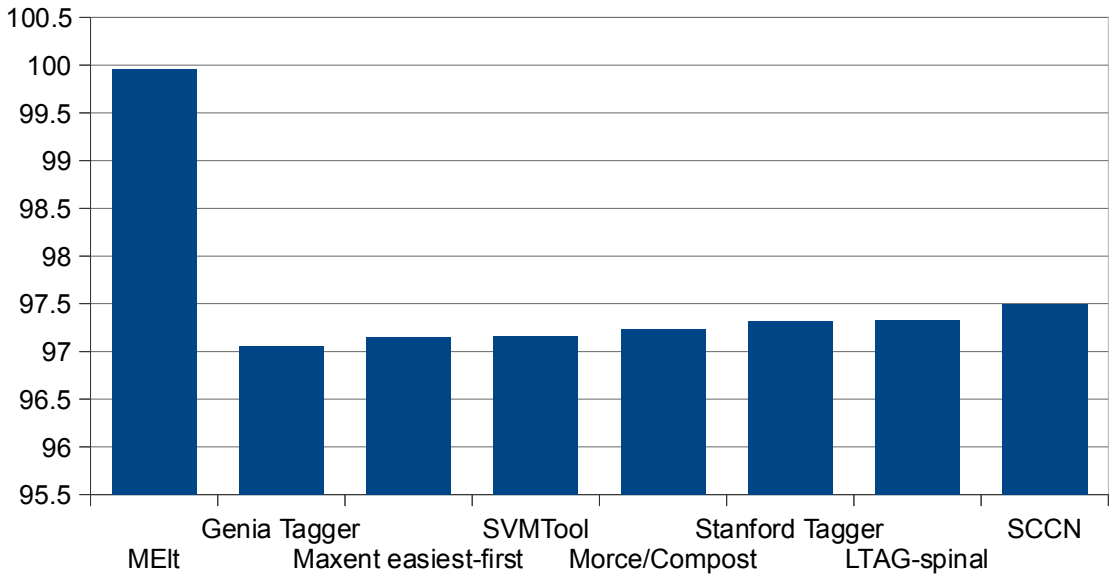
Sentence	POS-Tag
Let's have a talk	Noun
Let's talk	Verb

Nowadays, there are software pos-taggers that not only identify the part-of-speech of single words but also identify efficiently the grammatical tag of a word contained in a context.

There are two different categories of POS taggers: rule-based [9] [10] or POS Taggers based on stochastic markovian models [11]. Taggers which belong to the first group are efficient with formal texts which are following strict syntax rules. On the other hand, taggers that belong to the second category could adapt in not so strict environments. Moreover, they could apply efficient tagging in any word by finding the best match. This is feasible by having large training sets and calculating various probabilities about how likely is a certain word in a context to be a verb or a noun for example.
For the present research POS-Tagging is important in order to extract keywords from subtitles. The potential keywords should be nouns or nouns that denote names. In order to identify the grammatical type of a word, the POS-Tagger is a necessary tool. For the reason that subtitles are texts not following a strict syntax, a stochastic tagger is the best solution for the current research. There are various stochastic taggers and the most efficient among them are presented in the table below:

Tagger	Short Description	Accuracy
MElt	MEMM with external lexical information	99,96%
Genia Tagger	Maximum entropy cyclic dependency network	97,05%
Maxent easiest-first	Maximum entropy bidirectional easiest-first inference	97,15%
SVMTool	SVM-based tagger and tagger generator	97,16%
Morce/Compost	Averaged Perceptron	97,23%
Stanford Tagger	Maximum Entropy cyclic dependency network	97,32%
LTAG-spinal	Bidirectional perceptron Learning	97,33%
SCCN	Semi-supervised condensed nearest neighbour	97,50%

Table 1: This table presents the most popular POS-Taggers and their efficiency.



Graph 1: A graphical representation of the efficiency of most popular POS-Taggers

From the above table is concluded that the most efficient POS-Tagger is the SCCN. However, because of copyright reasons and also due to the fact that the difference in the accuracy is not very big, the Stanford Tagger is the most suitable for this project because it is both accurate and published under the GNU General Public License.

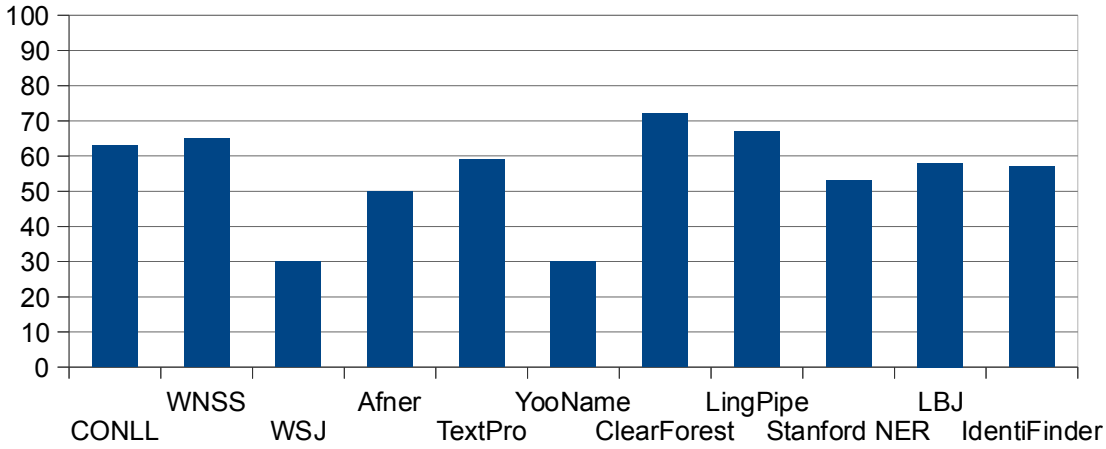
2.3.2 Name Entity Recognition (NER)

Name Entity Recognition is called the task of identifying names in a text [12]. This technology is widely used in text mining as well as for information extraction. Nowadays, there are various software packages that use NER techniques based on grammatical rules or machine learning techniques which efficiently identify name entities.

Typically, rule-based NER systems achieve better efficiency but they are slow because, the run grammatical rule checks for each word in a context. On the other hand, Machine Learning NERs need a large corpus of manually labelled training data. Manually labelling data is a difficult and time consuming process. That is the reason why the NER researchers try to find more effective ways to construct such systems. This is feasible by applying semi-supervised learning and extracting Names from Wikipedia.

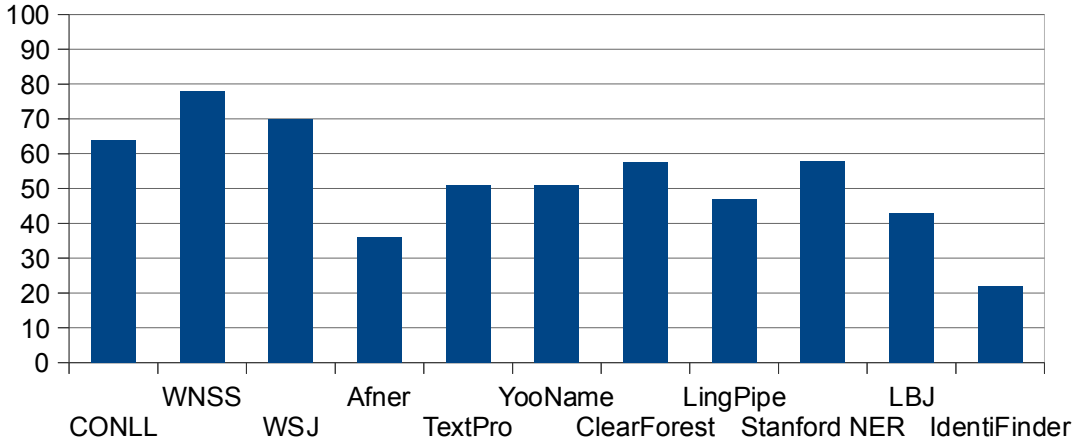
Below it is presented a comparison [13] [14] between various Named Entity Recognisers, both rule-based and statistical-based. Notice that for the present research, named entities from subtitles is needed to be extracted. These entities could be people names but also organisations' names or locations. Thus, the comparison is divided in three different groups of named entities. The comparison made with a large corpus of English articles.

Entity Type: Person



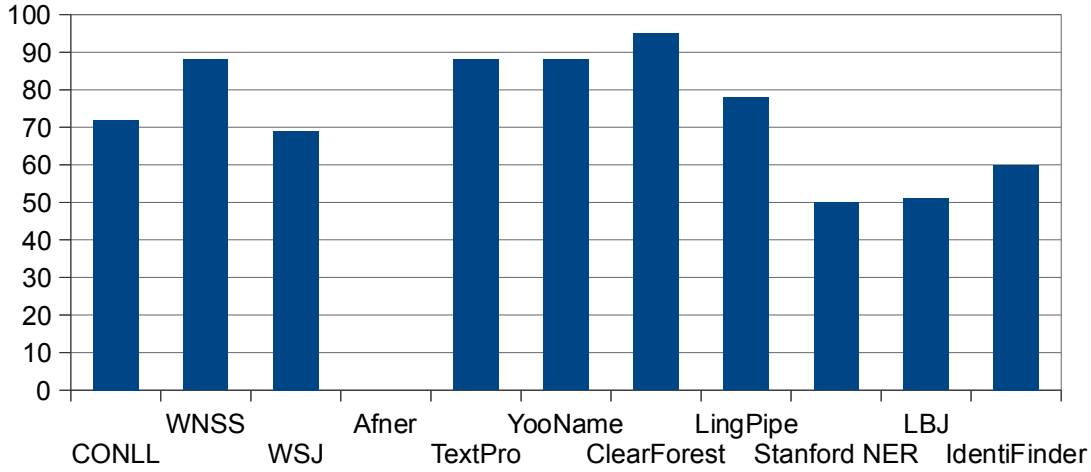
Graph 2: A comparison of different NER and their ability to identify person named entities

Entity Type: Location



Graph 3: A comparison of different NER and their ability to identify location named entities

Entity Type: Organisations



Graph 4: A comparison of different NER and their ability to identify organisation named entities

As it is clearly presented from the results, there is no NER outperforming all the others for all the entities tested. Each NER achieve high accuracy in a certain field. Although, ClearForest could recognise efficiently entities that concern persons and organisations but, it does not possess good efficiency in recognising Locations. On the other hand, WNSS is clearly the best recogniser for this task. All of these entities are important in order to identify tweets relevant to TV broadcasts and all of the named entities appeared in subtitles, are equally important. People names extraction is useful in order to extract names of the characters from TV-series. However, location entities are necessary as well because they reveal important information about the plot of an episode. It is clear that there is a trade-off what should be taken into account. NERs are not going to be used in our developed system. However, the NERs constitute a basic element of the existing system that SecondSync company is using. Furthermore, the NERs could be used in conjunction with the Tag extraction system as it will be explained in the suggested architecture chapter (2.4.2).

2.3.3 ORM and Hibernate

The SecondSync platform produces *real-time* TV analytics. As a result, time efficiency is a major concern for achieving the aims of the system. Though, this is really challenging because the filtering system that the present research develops, should be able to process large chunks of datasets in a limited time efficiently and simultaneously.

First of all, as described in the introduction chapter, large amounts of subtitle texts should be processed in order to apply keyword extraction techniques. This includes continuous transactions with a database such as the following:

- Subtitle retrieval
- TV-listings retrieval
- join database commands in order for the correspondence between subtitles and broadcasts to be found.
- Fetching Training Set consists of tweets, labels and features
- Keywords insertion which is a result of the keyword extraction phase

These transactions are expensive in a time-consuming point of view because, they require procedures such as conversion between datatypes. The ideal case would be to manage the datasets as java types/classes and not as database types that later should be converted in java types in order to be further processed. Furthermore, this would be more flexible because, we could create our own datatypes that could be directly imported in the database.

For instance, assuming that a trained set is available and is represented in one database table and 5 columns. The first column is a tweet, the second is the label of this tweet (relevant to TV broadcasts or not), the third is the exact time that this tweet had been published and the other 2 columns are the features that each tweet had achieved during the training. This table consists of 3 different types (Tweet->Text, Label->String, Time->Timestamp and Features->Boolean). The results that the database returns cannot be inserted in a java List with a certain datatype because, the results have different datatypes. As a result, all these different entities should be separated and converted in different typed structures in order to be processed.

On the other hand, if java types could be used during the transaction with the database only a custom datatype would be present, called “TrainingSet”. The “TrainingSet”would contain different internal fields including the information described above (tweet, label, time, features) and all the necessary functions to retrieve them. By using java objects in the transaction with the database, only one typed List is needed (the List contains “TrainingSet” objects only) containing Objects with all the necessary information. In order to achieve the best possible efficiency in the transaction with the database Object-relational mapping had been used.

Hence, Object-relational mapping is an intermediate layer that is used in a software system for converting information between different datatypes. This research uses the Hibernate software package in order to do improve the efficiency of the developed system.

2.3.4 WEKA

WEKA[15][16] stands for Waikato Environment, for Knowledge Analysis and is a tool which provides researchers and programmers with state-of-the-art techniques for data analysis, classification, data clustering, prediction models construction, data mining and other machine learning components. In more detail, WEKA is a package built in Java containing tools such as:

- Machine Learning Algorithms (Classifiers)
- Training Set Handling and Creation techniques (Datasets handling, tools for reading training sets from databases, features creation)
- Visualization Tools (Graphs)
- Data preprocessing Toolkits
- Evaluation Methods (Roc Analysis, f-Measure, Recall)

This open source software package is platform independent, as it is developed in Java and is accompanied by a user graphic interface. Furthermore, it is completely flexible as it could be used as a command line tool and that fact makes it accessible from any programming language. Last but not least, the major asset of the functionality that WEKA provides is that it facilitates the experimentation between different machine

learning algorithms.

2.4 Architecture

The architecture of the SecondSync platform is really important and should be mentioned such that the reader would be able to understand further the functionality of the system. Also, the architecture of the potential system suggested by the current research is going to be presented below. As a result, the reader could understand practically the differences in the two systems and read a complete comparison between the two systems.

At this point it should be mentioned that the current system produces real time TV analytics. As a result, the system should be constructed in such way that could be used in order to produce accurate results in real-time. This could be complicated because the current research is based on subtitle datasets that are complete. However, this is not feasible when producing real-time analysis. For example, when a broadcast begins we do not have in our possession the whole subtitles corresponding to this show but only a part of them.

In more detail, the subtitles data that is in our possession when we implemented our keyword extraction phase does not exist in the reality because, the system works in a real-time manner. Specifically, when a broadcast begins it does not exist as a complete subtitles dataset but only a part of them. It is only at the end of the broadcast that all the subtitles of the certain broadcast are known

Our system is not capable of doing this without having at least a fair amount of subtitles. As a result, a complementary system is required that will run at the beginning of the broadcast in conjunction with our tag extraction system. This could be a NER. Another concern is the efficiency of the system developed. As denoted by the company the keyword extraction phase must be able to run in less than 30 seconds. All these problems and their solutions are presented below in more detail.

2.4.1 Current Architecture of SecondSync platform

The current platform consists of three major components:

- Real-Time Named Entity extraction from subtitles
- Real-Time search terms Construction based on the named entities found on the first phase and twitter querying
- Classification of the results returned from the previous

The reader should think the complete structure as an open stream that provides the system with subtitles and tweets continuously. Each subtitle imported in the system, is filtered for named entities by using NER techniques. Afterwards, the system is using the extracted named entities to search for tweets possibly relevant to TV broadcasts, by querying the tweets database. The returned results are filtered by a general-scope classifier that identifies the desired tweets. The key-point is that this procedure is repeated continuously and as a result the system is able to produce real-time TV analytics. The structure of this system presented schematically below:

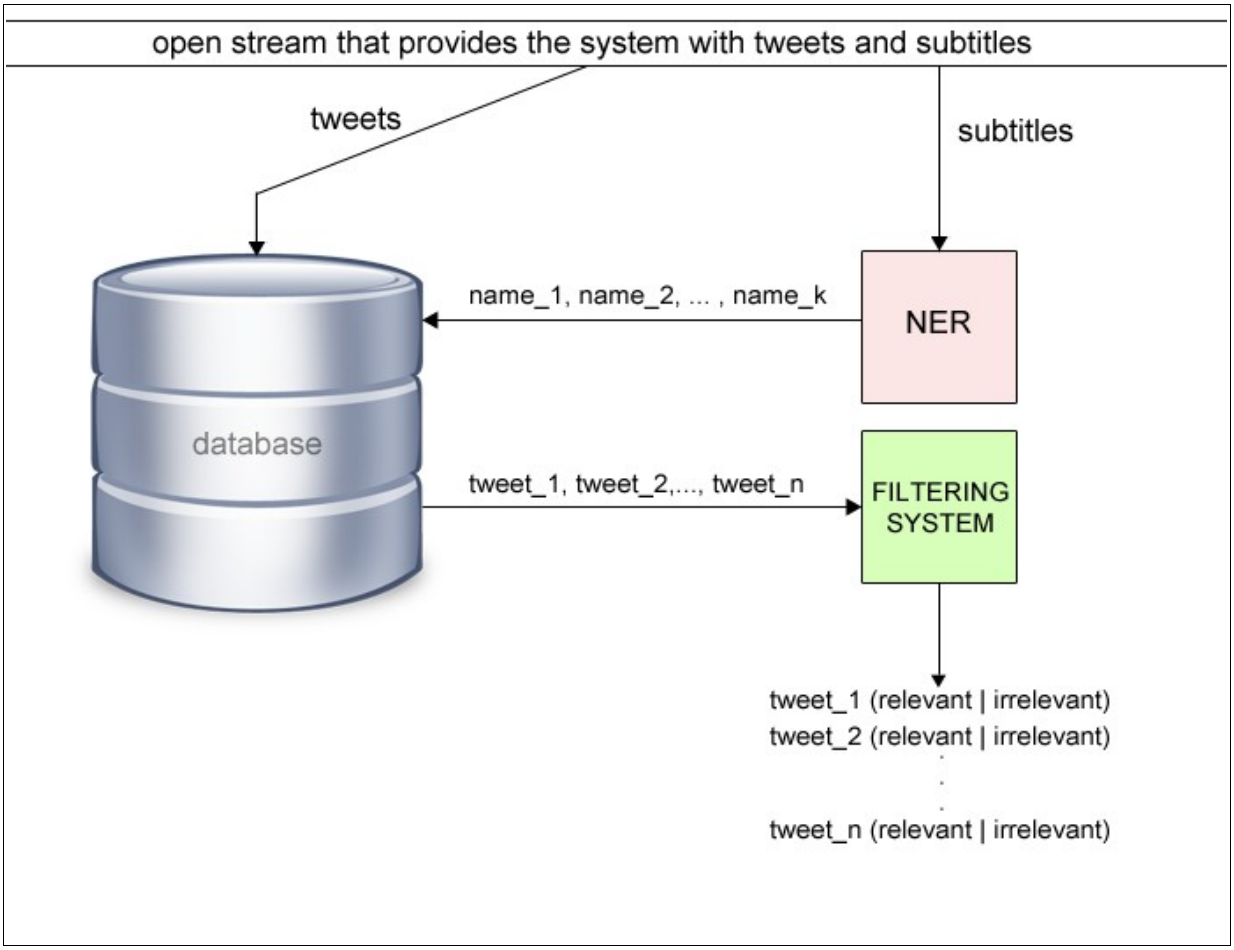


Figure 1: The architecture of the system that SecondSync platform uses.

Advantages of this system:

- ***Fast subtitles processing.***
  Each subtitle is filtered for named entities only and not for other keywords that may describe the TV-series. This is a straightforward technique and does not demand additional processing. As a result, it produces results quickly and facilitates the real-time processing of the datasets (subtitles).

- ***Fast filtering.***
  The system has not a specialized training set but is using an already compiled set which may not have been trained only for TV-series. As a result, the system is not responsible for updating the training set with new entries. Moreover, because of the fact that the filtering system is not checking each tweet for different specialized features, less time is required for the completion of the process. On the other hand, a filtering system with custom specialized TV oriented features would be more time consuming.

Disadvantages of this system:

- ***The keywords extracted from subtitles are named entities only.***
  This fact reduces the set of the search terms that the system is using to retrieve tweets. In fact, the tweets returned, are only those containing names of the actors

or the title of the broadcast. This is the case only if the named entity recogniser is able to identify correctly all these terms in the subtitles. The ideal would be to track not only names but also words or phrases which are continuously repeated during the broadcast. Furthermore, sometimes NER techniques do not guarantee that all the named entities are recognised. Also, some misspelled words that appears in the subtitles could be identified as names by the NER.

- ***The classifier used is to generic and it is not TV-oriented.***
  The efficiency of the whole system is affected by the above fact and as a result many false positives are produced. In more detail the filtering system returns tweets that is decided from the classifier that are relevant to TV broadcasts but actually they are not.

2.4.2 Potential Architecture of SecondSync platform

The system that the current research suggests consists of four distinct components:

- Keyword Extraction by using the subtitles dataset and TFIDF[1] measures. TFIDF techniques are used in order to identify important words in a certain context. Normally, this is done by counting how many times a word appeared in a text. If this word is not repeated a lot of times in other texts it means that it is indeed important for the current text.

- Named Entity Extraction for each subtitle imported in the system. This mechanism is already built by theSecondSync and will not be examined in the present research. However, a combination of Named Entity Extraction and Tag Extraction based on TF-IDF measure could produce accurate results.

- A structure that combines the above systems and make them able to produce keywords describing the current TV-series/Movie in real-time. The keywords consist of names and other keywords as well.

- Real-Time search terms construction by using the keywords collected from the above tasks.

- A filtering system that consists of a special-purpose classifier. The classifier is using features that are directly linked to TV-series and Movies. These features are described in more detail in the 4th chapter.

- This component is completely independent from the above components. Past subtitles are used in order to statistically analyse all the contained words.

The current research should follow the same regulations as before. The general platform is an open stream providing the main system with tweets and subtitles continuously. The tweets are inserted in the database as well as the subtitles. However, the newly imported subtitles are analysed by two different components. There is a Named Entity Recognizer extracting named entities from the subtitles and a system using TFIDF techniques in order to extract keywords (both names and nouns) from the subtitles. Subsequently, all these keywords will be used in order to query a tweets database and return a potential

1 TFIDF is a statistical-based technique for extracting keywords from documents. It will be explained in more detail in the next chapter.

set of messages relevant to TV broadcasts. Finally, the returned tweets will be filtered by a Bayesian classifier (see chapter 4) in order to separate the messages which are indeed relevant to TV broadcasts. The classifier is based on custom features compiled from characteristics appeared more often in tweets relevant to TV-series/Movies.

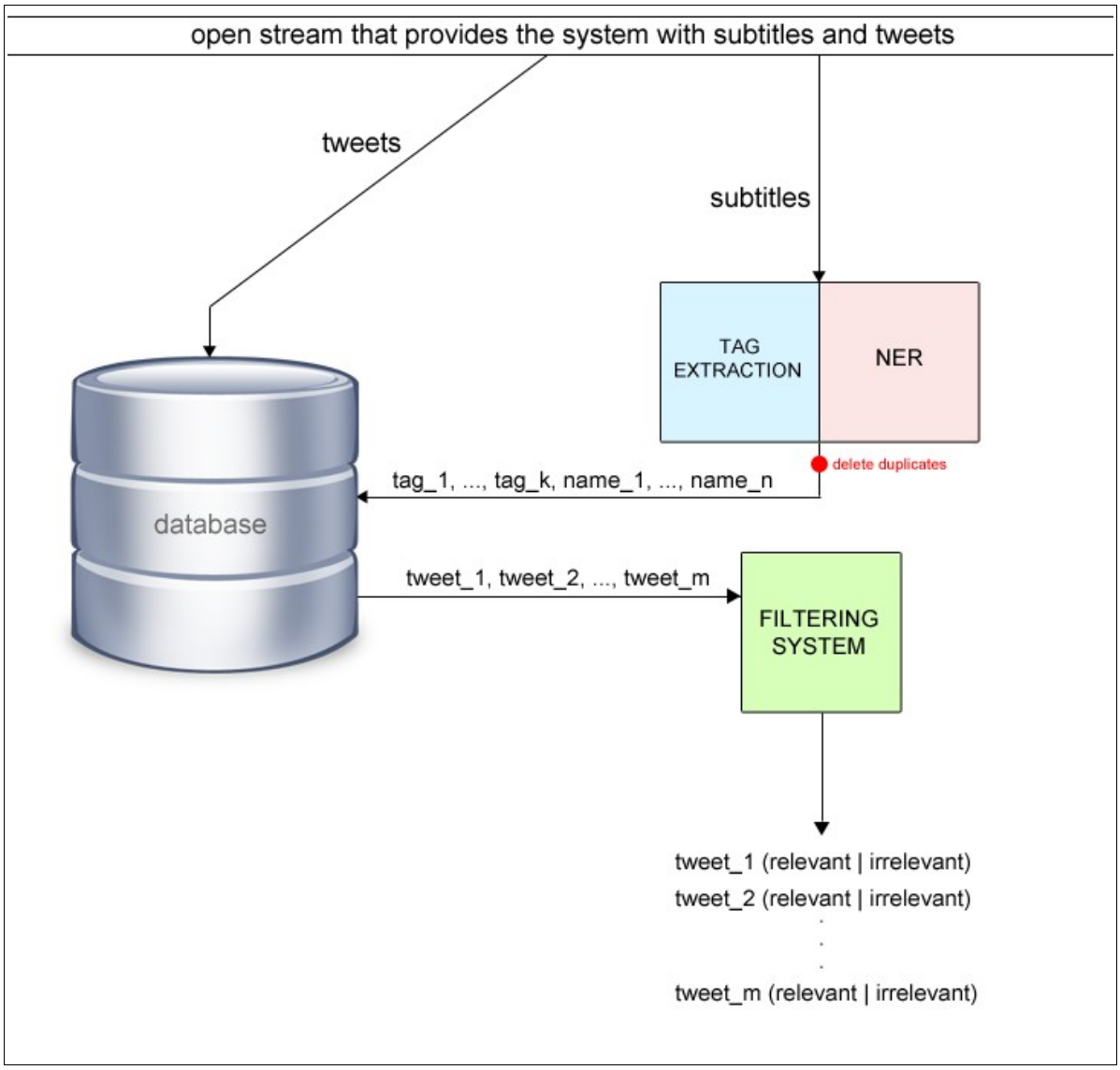


Figure 2: The architecture that this research suggests for the SecondSync platform.

This architecture gathers numerous advantages:

- ***More keywords are extracted.***
  This is a major advantage because, besides the fact that these keywords are directly linked to the broadcast, more search terms could be constructed as well. By using these terms a bigger set of tweets that might be relevant to the current broadcast is returned.

- ***TFIDF mechanism validates and correct problems in NER's functionality.***
  Some names appeared in the subtitles are not efficiently identified by NER techniques. This is because NER's do not have perfect accuracy. If these names are important in the plot of a TV-series and are repeated many times in the subtitles. Our keyword extraction mechanism is able to identify such names. As a result, the failure of NER to successfully identify some named entities is balanced

by the TFIDF mechanism.

- ***Error Correction because of fault in the subtitles.***
  The subtitles are generated automatically by Speech-To-Text technologies and as a result there are numerous faults. In such occasions and if the fault is in the beginning of a sentence the Named Entity recognizer identifies the false words as named entities. The experiments made with the subtitle datasets revealed this case. For example, if the false word “Iam” (“I am”) appears in the subtitles the Named Entity Recogniser would tag this as name which is wrong.

- ***Reduced false positives.***
  By constructing a classifier with custom features, a better accuracy had been achieved.

Besides all the advantages of this architecture there are some major disadvantage:

- ***The TFIDF technique cannot be used as standalone keyword extraction method in real-time.***
  In order to produce keywords, the TFIDF technique, whole the amount of the subtitles should be taken into account. However, if the keywords in real-time are desired to be extracted, it is impossible to do so without taking into account the whole set of subtitles However, this is impossible because if we want to extract keywords in real-time we should do it without having the whole set of the subtitles. For that reason the current structure is running a NER and a TFIDF mechanism, simultaneously.
  More information about this problem and its solution is presented in the following chapter where the TFIDF technique is described in more detail.

- ***Additional Complexity.***
  The TFIDF system adds additional complexity to the system.

2. Tag Extraction from Subtitles

As had been mentioned in the introduction chapter, one of the major aims of this project is to extract terms that will be used in order to search the tweets database for tweets likely relevant to TV broadcasts. The current method that the company is using to do so is by hash-tagging the title or the channel of the broadcast and query the tweets database. Instead of searching tweets, only by taking into account the title of the broadcast or the channel transmitting the TV-series, the present research is exploring a more advanced method for producing tags from the subtitles of the TV-series. In more detail the designed mechanism is exporting tags that are names of characters or terms describing facts happened in the current episode. Subsequently, these terms could be converted to search terms in order to query tweets database.

The intuition here is to expand a set of tweets returned by the previous methodology (searching by titles only) and compile a larger set of tweets that is potentially relevant to TV broadcasts. The current set will be definitely larger because, more search terms had been constructed, and so will return more tweets. This is an asset because the aim of the SecondSync platform is to find as much as possible number of tweets. This fact will likely produce not only more tweets in general but also more tweets relevant to TV broadcasts.

The present chapter deepens the methods that are going to be used in order to increase the number of the search terms and this is feasible by using the subtitles. In more detail by applying to them Natural language processing techniques. Here we examine the various methods that all together are widely known as Natural Language Processing (NLP). NLP includes procedures such as:

- Preprocessing and Tokenization [17]
- Tokens Tagging and candidate keywords Extraction
- TF-IDF analysis
- Tag Extraction

In order to link all the above together a small and high-level description of how Natural Language Processing works is the following: First of all, assume that a large corpus of documents is present and keywords should be extracted from one of them. This document is given as input into a preprocessing mechanism that separates the whole text into single words. Afterwards, a part-of-speech tagging is applied to the words (tokens) in order to be decided their part of speech. Usually only nouns or names are considered as potential tags. After the part-of-speech separation a TF-IDF technique is applied in order to extract important terms for the document. Such techniques extract tags by considering the frequency that a word has in a context. Finally we define a limit for how many tags we want to extract and our Tag Extraction methods outputs them. It should be mentioned that this method is based on statistics and NLP.

Apart from this method Machine Learning techniques could be used for extracting keywords from subtitles with many [21] [22] known mechanisms such as the KEA algorithm [23], which is the most popular one. At this point an argument should take place in order to indicate why NLP methods and Statistics had been used for the development

of the system and not methods based on Machine Learning Techniques.

3.1 Why not doing Tag Extraction based on Machine Learning approaches

Kea is using the Naïve Bayes classifier for training data[23] and extract key-phrases from a given content. The two stages of the algorithm contain the training of the data and the key-phrase extraction. During the first stage, a huge corpus of documents is necessary in order to accomplish a well-formed training. The documents contained in the training set (TS) have already some descriptive keywords attached. These keywords have been manually extracted from the documents by their authors. The next stage is using the above model in order to extract key-phrases for a not-previously-seen document. The complete functionality of this algorithm is presented in the following figure as seen in [23].

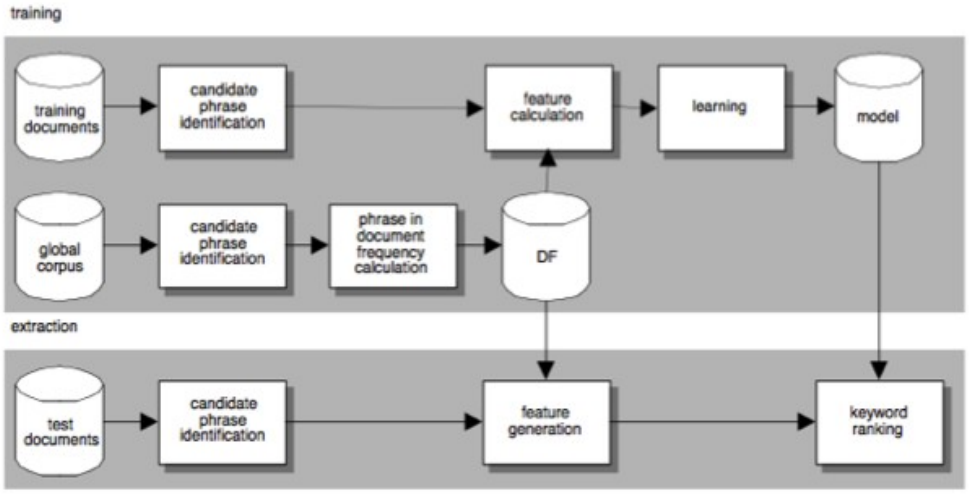
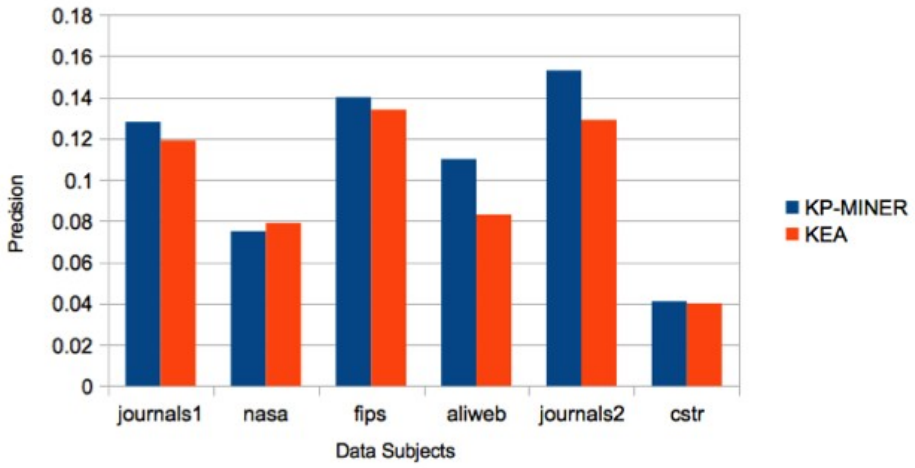


Figure 3: Architecture of ML Tag extraction systems.

The next step is to make a comparison between the most popular ML extraction method and one of the most popular NLP extraction methods, known as KP-Mining. According to the experiments that took place in [24], the following chart represents the precision that each algorithm achieved after the automated extraction of twenty tags in total.



Graph 5: An efficiency comparison between KP-MINER and KEA algorithms.

This comparison is clearly illustrating that the algorithm KP-MINER is based on statistical

methods outperforms KEA. Furthermore, the same study states that the KP-Miner also outperforms another ML extraction system known as Extractor[21].

As it seems in the most of the cases, statistical NLP methods outperform Machine learning NLP methods. Now it is clear that the first aim of this research concentrates on the fact of constructing an efficient NLP method that suits subtitles. Another reason which reinforces that choice is that in order to use algorithms based on machine learning techniques, a massive amount of documents already tagged with keywords would be necessary. More clearly, a training set should be compiled. This would contain a huge amount of articles which are manually tagged with keywords. These articles could be related to economics, politics, society and several other subjects. Machine Learning algorithms could be efficient only for tagging texts, related to the corresponding subject. However, the current research does not concentrate on tag extraction from subtitles that concern information from specific data categories such as economics or politics. As known, subtitles extracted from TV series could contain several different types of information about any topic, thus ML tag extraction algorithms do not give a solution to the problem.

After arguing that the most suitable method for extracting tags is NLP based on statistical methods , all the necessary stages of this methodology will be described in the following chapters. However, the TF-IDF measure that this technique uses should be explained as well as it's importance for the current research.

3.2 TF-IDF Measure

TF-IDF is a statistical measure indicating the significance of a token in a certain context or corpus of documents. The intuition here is this: the more important a token is for a context, the more times it appears in the specific context. However, there are some words like “is” or “do” that despite the fact that their occurrence probability is high, they are not so important in a context. TF-IDF measure takes this fact into consideration. Finally, the TF-IDF metric produces a probability that denotes the importance of each token into a document. Here should be noted that almost all the keyword extraction algorithms[29] are using the TF-IDF measure in order to extract keywords. Mathematically TF-IDF score is described as:

$$TFIDF = TF * IDF$$

Below, is explained further which tf and idf score are explicitly and why they are important.

3.2.1 TF

TF stands for Term Frequency. It is a mean representing how many times a token (word) is displayed in a document. TF defines the importance level of this word in the current document. However, using only the TF factor for measuring the importance of this word would not produce the expected results. This is because of the existence of common words such us “is”, “and”, “it” which will always have high level of importance if taking into account only the TF Factor. In order to avoid similar situations, the IDF factor had been introduced.

Furthermore, it might be thought that a certain term may appear more times in a larger document than in a smaller one and as a result the TF score is not so accurate. The

solution to this problem is called normalization. More specifically, in order to normalize the TF value the portion of the term frequency of the desired token is calculated with the largest number of occurrences of any token in our context. Mathematically it is described as follows:

$$tf\left(t\,,d\right)=\frac{tc\left(t\,,d\right)}{max\left\{tc\left(w\,,d\right)\right\}}\quad,\quad\left(t\,,w\right)\in d$$

3.2.1 IDF

IDF stands for Inverse Document Frequency. It is a mean which represents how many times in total a word is appeared in a set of documents. This score is important because it balances the phenomenon that some common words appeared in a regular basis related to others. IDF is defined mathematically as:

$$idf\left(t\,,D\right)=\log\frac{\left|D\right|}{\left|d\in D\right|}\quad,\quad t\in d$$

here D denotes the total number of the given documents and d is the number of the documents where the term t is appeared.
After the theoretical presentation of the important for the current research TF-IDF score, the exact tag extraction methodology is presented.

3.3 Tag extraction based on NLP and Statistics

3.3.1 Preprocessing

Preprocessing is an important phase in the present research and is related to the datasets format that are in our possession. As mentioned above, the subtitles dataset will be used in order to produce tags for each desired TV-series. However, this is not a simple task. The difficulty concentrates on the fact that the subtitles dataset is not clean but contains a lot of noise. The noise involves:

- Meaningless words produced by errors from the Speech-To-Text procedures (chapter 2.1.1)
- Words not suitable as keywords (verbs, adjectives etc.)
- Punctuation

Apart from this, the subtitles should be compatible with the developed system. As had been proposed in [20] the preprocessing phase consists of four different stages:

- Data cleaning
- Database normalization
- Data Transformation
- Feature Extraction

3.3.2 Data cleaning

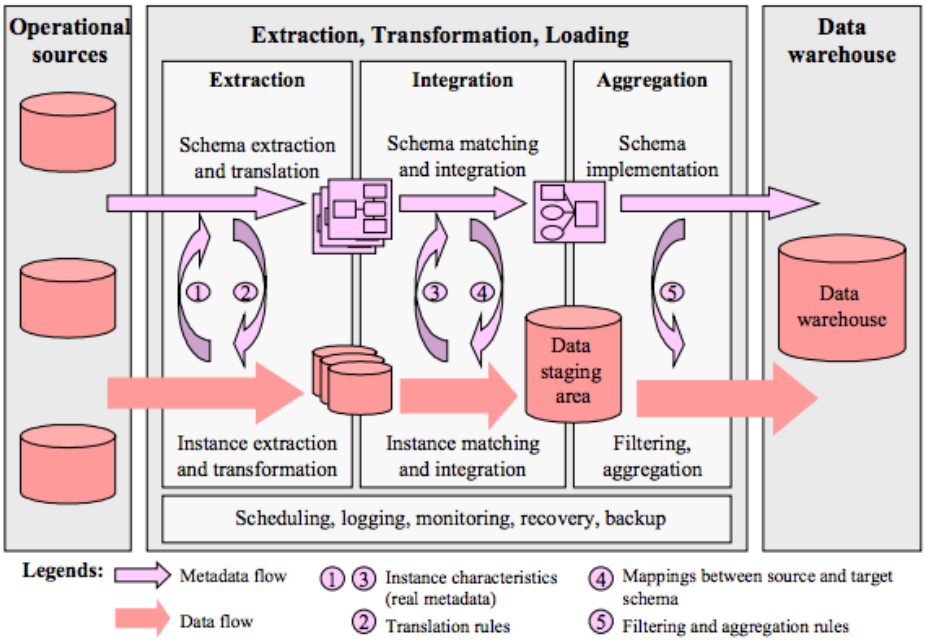


FIGURE 4: Summarisation of the cleaning procedure that every system should follow before using datasets *

Data cleaning or scrubbing is called the phase of preprocessing focusing on data transformation and data error detection as well as correction. The aim of this procedure is to produce quality datasets that could be analysed without further problems such as data inconsistencies. For the tag extraction system only the subtitles content is necessary and the TV-listings subtitle. Before using them we should address the problems that may be appeared in the given datasets:

Problems:

- *Invalid format*
  The aim is to produce tags from meaningful text that describe perfectly the subtitles. The desired keywords are sequences of letters or numbers and there is no reason for any punctuation to be contained in the subtitle dataset. The following table contains symbols which should not be contained in the desired "clean" version of datasets. Moreover, after analysis of the subtitles, some sequences of symbols had been found to appear many times in the context. Possibly because of errors during the generation procedure (note that subtitles are generated from text-to-speech technologies). These sequences should not be contained in the clean version of subtitles.

SYMBOL/SEQUENCE	DESCRIPTION
.	period
;	semicolon
,	comma
...	ellipses
:	colon

* Data Cleaning: Problems and current approaches. Erhard Rahm, Hong Hai Do, University of Leipzig, Germany

*	asterisk
-	dash
() or []	Parentheses or brackets
/	virgule
!	Exclamation point
?	Question mark
#	Hash Tag
%	percent
_	underscore
“”	Quotation marks
_ _	Double underscore
_ - -	Unrecognised word/phrase by the subtitles generation system
--	Unrecognised word/phrase by the subtitles generation system
“ ” or “ “ etc.	Sequences of blank spaces.
/n	New line characters

In the research all these situations had been located and transformation methods had been applied to datasets. This was feasible by applying complex expressions matching the undesired patterns or symbols. As a result, all these patterns could be successfully identified and removed from the subtitles dataset. These techniques will be presented in more detail later in this section.
As far as the TV-listings are concerned, they were provided in text format. This was not convenient for our research because, the start/end time of each broadcast was also in text format. This makes comparisons between dates infeasible. As known, programmatically, it is completely different to compare text and dates. As a result, transformation to this dataset had been applied converting the date entities, from text format to the Date java class datatype.

- ***Errors in datasets***
  The subtitles dataset contain a lot of misspellings due to the mechanism generating them. Speech-to-Text technologies nowadays are quite efficient but not perfect and as a result they produce errors.

- ***Missing Information***
  In order to apply accurate analysis in large chunks of data, these should be complete. However on many occasions this is not feasible. Usually, a sub-system is necessary in order to supplement these datasets by finding additional information on internet or another databases. In this way the consistency of the final data is guaranteed.

  Here we should notice that the subtitle datasets which are used in our tag extraction phase are complete so, there is no need of dealing with missing information at this stage. However, in order to assign subtitles to TV-series there is the need to know about when these TV-series were aired during the day. This information had been searched on web by SecondSync company and the owners provided this research with this information (TV-listings). Therefore, the missing

information problem had been already handled by the SecondSync technical team so we did not have to do additional data mining for this task.

All the problems listed above occurred during the analysis of the datasets. These datasets had been provided by the owners of the SecondSync platform in format that wasn't processable from our tag extraction system and was containing errors as well. Thus, the subtitles were transformed and corrected in order to become a valuable high quality dataset. Therefore, the solutions that had been applied to the datasets in order for the above problems to be solved are listed below.

Solutions:

- ***Invalid Format***
  As far as the subtitles datasets is concerned regular expressions had been applied in order to remove undesired patterns or symbols.
  Regular Expressions [25] [26] are grammatical expressions matching strings of a text. The strings could be sequences of symbols, letters or numbers. Regular Expressions are commonly known as regex or regexp and are based on automata theory. Regex could be created in a formal language, in every programming language. A preprocessor is used in order to match the pattern constructed in the regex language to a real world text. Below are presented some examples of regex matching strings with specific format:

REGEX SPECIFICATION	STRINGS MATCHED
[A-Za-z]+	Every word compiled by small or Capital letters e.g ReGuLAr, regular, REGULAR, ReGuLaR
[0-9]+	Every number e.g. 125, 5, 1856, 0000
[\\n]+	Sequences of newline character

Now that the theoretical part of regular expressions had been explained, the problem of how this research managed to remove the punctuation and the invalid sequences of symbols appeared in the subtitles dataset is described as well. First of all, a list with all the subtitle sentences contained in the database was retrieved. For each one of them the following Regular expressions had been compiled and applied in a strict row in order to remove all the undesired patterns.

REGEX SPECIFICATION	DESCRIPTION
[/\\!:()*._`,;?\"<>|]+	Match every punctuation symbol that contained in the subtitle string
(-)+	Match every sequence that contains multiple dash characters
[\\n]+	Match sequence of newline characters
's	Match the possessive 's token

By applying these regex expressions, all the undesired patterns had been matched and then by using the "String.replace" java method all the matched patterns had

been replaced by a blank space. This task produces sequences of blank spaces when two or more matches occur. However, the desired format of the subtitles dataset contain numbers and words separated only by a singe blank space. The consecutively blank spaces had been matched by the following regular expression:

REGEX SPECIFICATION	DESCRIPTION
( )+	Match sequence of blank spaces

Finally the same java method as before had been used in order to replace the sequences of blank spaces, that may occur, with a single space. This step produced a subtitles dataset consisting of words and numbers separated by a single space character.

- ***Errors in datasets***
  As mentioned above the subtitles dataset contain a lot of misspellings due to the speech-to-text mechanisms used to generate the subtitles in real time. Some of the most common mistakes present in the dataset are:

Meaningless Word	Consists of
Iam	“I” “am”
Iplay	“I” “play”

Such errors occur only in the beginning of a specific sentence creating a lot of problems in the current system that SecondSync is using. In fact, as had been mentioned, the existing platform is using NER software to extract names from titles. As a result, this system will possibly identify wrong examples displayed above as named entities producing inaccurate keywords. The current study resolved this issue by following the above straightforward algorithm which takes as input a word that is possibly a name:

1. Check the word that is possible a name and locate any capital letters.
2. If the first letter is capital and all the other characters are not then continue or else exit.
3. Remove the subject (e.g I, you) and check the remaining token by using a POS-Tagger.
4. If it is a verb then it is indeed an error.

This procedure reveals some of the existed errors in the subtitles. As mentioned before the tags that had been extracted are nouns or names and as a result removing the above types of errors from our datasets does not affect the whole procedure. After this step the subtitle dataset consists of words separated with commas and without the errors that were described above.

3.3.3 Database Normalization

Database Normalization is the procedure of formatting the database scheme in such way that minimum redundancy is produced. Furthermore, this procedure assures that the database tables are completely independent and changes occurring to one table do not

affect the quality of the other tables in the schema.

A more straightforward description of why database normalization is important is this: Assume that in our project there is a web crawler that searches for TV listings online. After crawling many sites, the crawler accidentally inserts in the database, two different airing times for the same TV-series. Because of this error, the sub-system that crawls subtitles for each broadcast that exists in the TV-listings table, would insert to the database two different sets of subtitles and among them one that does not meet reality. This error would produce data inconsistency and taking into account that our system is based on statistical methods, some fault results could occur because of that. On the other hand, if somebody had applied database normalization procedures in the TV-listings table, this error would not have occurred. Furthermore, another inconsistency could occur in the case where the TV-Listings crawler inserted the TV-series title, in the TV-listings tables despite the fact that it could not find the exact airing time. In this case the TV-listings table would contain null records which affects negatively our system's construction.

Considering all these, it is clear that this step of the preprocessing task is very important. As a result, this task was accomplished by making a thorough check to the datasets about null records or duplication. The datasets were very consistent (apart from some events of duplication) and the suggested schema by the company avoided redundancy and dependency. Furthermore, some checks made in order to reveal inconsistencies such as wrong airing time for the examinable TV-series. In more detail, the examinable TV-series were cross-checked by finding their airing time on the BBC website. Furthermore, it had been checked if the subtitles which were extracted between the airing time of each TV-series, were indeed the subtitles of the corresponding series.

3.3.4 Data Transformation

Data transformation is the preprocessing task of transforming datas from one format to another, in order to be compatible with the system. As far as this research is concerned, this task includes the transformation of the datasets from the initial format to a format that suits the developed tag extraction system. The datasets were given to us as CSV structured texts and were stored in a MySql database.

To begin with, the CSV texts were fed to the java program. The text files are transformed to java datatypes (Lists of subtitles and TV-listings) by using an CSV Reader and afterwards by using the Hibernate tool as an intermediate layer. Hibernate helps to map the java datatypes to a certain MySql schema or MySql datatypes(see chapter 2.3.3). This is a single procedure which was made only once when the datasets had been provided and imported in the developed system.

However, extracting tags from certain subtitles, the necessary datasets should be fed to the tag extraction system. The system is implemented in java. Thus, the subtitles and TV-listings dataset should be inserted to the extraction mechanism in a format that java supports (Lists, ArrayLists etc.). These datasets are retrieved from the MySql database and then transformed from MySql datatypes to java datatypes by using Hibernate.

The following graph illustrates the whole procedure:

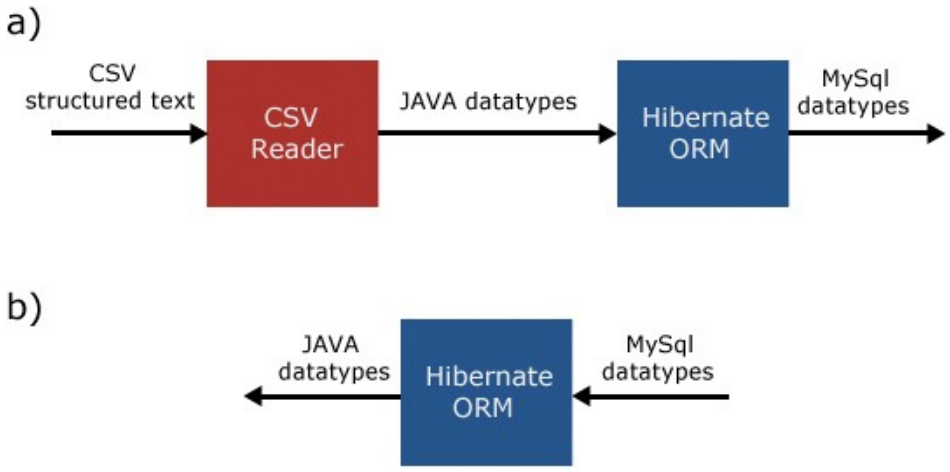


Figure 5: Conversion between different datatypes in the developed system.

Hibernate is the intermediate layer which is responsible for handling all the necessary conversions between datatypes. This layer not only facilitates the whole procedure but also contributes to the efficiency of the tag extraction mechanism.

3.3.5 Feature Extraction

So far, the suitable transformations had been made in order for the system to be able to process the data. Furthermore, some errors present in the datasets had been corrected and the consistency of the data had been assured. In addition, the subtitles records had been already collected in java datatypes (Lists) in a clean format (only words separated with a single blank space). Providing these, the next step of the research is to extract some features allowing to extract tags. More specifically, the next step is to produce a reduced representation of datasets suitable for feeding it into the tag extraction system. This representation will be a java List containing all the words appeared in the whole collection of subtitles. Furthermore, a list containing all the words appeared in the subtitles of the given TV-series should be compiled as well.

In order to create these List from the subtitles records a procedure known as tokenization was followed. Tokenization is a preprocessing task for applying NLP techniques to text Contexts. In more detail, this technique is responsible for breaking a string input into meaningful tokens or atoms[8]. Tokens are words or numbers. More specifically, we work as follows:

1. The system retrieves all the subtitle sentences in a clean format (words/numbers separated by a single blank space) and adds them in a List. Each entry/sentence is a string containing multiple words (tokens).

2. Our system applies the function String.split() to each distinct subtitle entry of the List. This function returns an array with the words (tokens) contained in the string.

Subtitle Entry	Array after applying the split function
“This is the first subtitle”	{“this”,”is”,”the”,”first”, “subtitle”}
“This is the second subtitle”	{“this”,”is”,”the”,”second”, “subtitle”}

3. All the returned tokens are inserted into a new List that created at the start of the tokenization stage.

4. The above procedure is applied to the whole collection of subtitles. However, the same procedure had been followed for the subtitles of the selected TV-series from which tags wanted to be extracted.

5. Finally two Lists have been compiled. The first List contains all the tokens for the whole collection of subtitles and the second List contains the tokens extracted from the subtitles of the selected TV-series

These two lists of data are the final products of the preprocessing phase. It is useful to name these lists as they will be recalled in the following chapter:

$$ListA: \text{List with all the tokens extracted from the Subtitles Collection}$$

$$ListB: \text{List with all the tokens extracted from the subtitles subset}$$

The next step is to further process lists in order to find potential keywords. As mentioned before, the potential keywords could be only named entities and nouns. Each token could be easily tagged in both lists by using the Stanford POS-Tagger.

3.3.6 Token Tagging and candidate keywords extraction

Token tagging is the phase that will reveal the potential keywords evaluated by the next stage, which is the TF-IDF analysis. In more detail a POS-Tagger had been used to tag all the tokens for both the lists produced from the previous task and the candidate tags had been extracted. Specifically, both of the lists had been updated by erasing all the words whose part-of speech is not noun (singular and plural) or noun that denotes name (singular and plural). Both the updated lists, which contain only nouns and names, are fed to the next and final phase which is TF-IDF analysis.

3.3.7 TF-IDF Analysis

The TF-IDF analysis is applied to each token in the updated $ListB$ in order to link each one of these with a specific score. The $ListB$ is helping to be found the TF score (how many times a token is appeared in the List) and the bigger $ListA$ list is helping to calculate the IDF weight for each token of the $ListB$.

The challenge here is that the TF-IDF measure can not be applied directly because the tokens existing in the lists are not extracted by documents (TF-IDF widely used in tag extraction from documents) but from subtitles. The reason is that if recalling the IDF formal type, somebody could observe that a number of documents are needed in the corpus in order to calculate the score. These documents are texts with logical borders but, subtitles are sequences of sentences without logical borders meaning that have no start or end point as the documents do. As far as the subtitles are concerned, they could

be logically separated if considering the following intuitions. However none of the above ideas are very efficient:

- Each different subtitle in the database should be considered as document.
  This will not produce good results as it is difficult to find important words in such a small context. (each subtitle consists of roughly 10 words)

- All the subtitles of a certain TV-series/Movie should be considered as documents.
  This would be a suitable solution but the efficiency is very important for this research but is time consuming to make this separation for all the TV-series/Movies.

As a result the IDF mathematical type can not be used as it is but slightly amended. Specifically, The following solution came up:
Instead of using the total number of documents $|D|$ and the total number of documents containing a given token $|d|$ in the IDF measure, the total number of occurrences of a given token in the whole corpus of the subtitles dataset and the total number of occurrences of a given token in the subtitles of a certain TV-series/Movies had been used respectively. Mathematically, the amended IDF score for a token is presented below:

$$\bar{idf}(t, S) = \frac{|tc(t, s)|}{|tc(t, S)|} \quad , \quad s \in S$$

where s is the subtitles of a specific TV-series/Movie and S is the whole collection of subtitles. Furthermore, $|tc(t, S)|$ is the total number of occurrences of the token t in the set S and $|tc(t, s)|$ is the total number of occurrences of the token t in the subset s .

As far as the TF score is concerned, it is clear that there is no reason to use the normalization procedure because, all the subtitles have roughly the same size and there is no reason for additional calculations. So, the TF formula which had been used is presented below in order to calculate the TF score:

$$\bar{tf} = tc(t, s) \quad , s \in S$$

Finally, the $tc(t, S)$ and $tc(t, s)$ had been calculated for each token in the lists $ListA$ and $ListB$ respectively. The token and the corresponding values are inserted into the HashTables $HashA$ and $HashB$. The size of the hashTables are smaller than the size of the Lists because, HashTables do not allow duplicate entries and as a result every token in the Table is unique. The reader could imagine this as two vectors. The first one contains how many times each token is appearing in the whole subtitles collection and the other vector is containing how many times each token appears in the subtitles of the current TV-series.

Last but not least, an argument about what is the asset of using HashTables for storing all these tokens and not any other data structure should be made. As mentioned above, NLP needs a huge amount of processes related to tokens. These tokens should be

stored in a structure offering low complexity in actions such as token searching/insertion/deletion. An additional reason that forces this research to examine the efficiency of the selected data structures is that the system will be used by the company in a real-time manner (remember that the TV analytics that SecondSync platform produces are processable in real-time) and should export results as quickly as possible. Below is presented a table with various java data structures and their complexities:

Structure	lookup	Insert	Delete
Array List	O(n)	O(1)	O(n)
HashTable	O(1)	O(1)	O(1)
Stack	O(1)	O(1)	O(n)
Vector	O(n)	O(1)	O(n)
Linked List	O(n)	O(1)	O(n)

Table 2: Efficiency comparison for various data structures.

As may be observed, the best data structure for our system is the HashTable. The reason is that it achieves O(1) complexity for all the actions that the current research concerns. Of course, this is the case only if there are no rehashes. However, the rehashes are avoided because the used HashTables have bigger size than the number of the tokens going to store. Despite the fact that the exact number tokens is not known a rough guess had been made, as the sizes of $ListA$ and $ListB$ were known. Taking into account that the corresponding HashTables do not contain token duplicates as the Lists do, it is clear that their size would be smaller.

3.3.8 Tag Extraction

The previous step produced two HashTables with the term frequency for each token attached. This information allows the calculation of TF and IDF weights for all the tokens existing in the $HashB$. Depending on the $TF*IDF$ value that each token has, it could be said how important this token is for the chosen TV-series/Movie. Specifically, tokens with high TF-IDF score are potential keywords. The only decision that should be made is about the lowest TF-IDF value (threshold) that a token should have in order to be considered as definite keyword. It should be clear that a lower threshold produces more keywords.

Apart from this threshold, the TF score of each token itself is used as second threshold and its aim is to limit keywords that might be mistaken or misspelled words. For example, a word appears only one or two times in the subtitles of the broadcast appears and only one or two times in the whole collection of the subtitles (because of the fact broadcast's subtitles constitutes a subset of the whole collection of subtitles). This token has big IDF weight, fact that increases its TF*IDF weight and makes it a potential keyword, with high probability. However, tokens that belong to this category are actually mistaken words produced from errors of the Speech-To-Text subtitle generation procedure. Hence, by using this countermeasure the system considers as keywords, only words that have high TF*IDF weight (first threshold) and they have appeared in the subtitles of the given broadcast more than two (second threshold) times. This technique

ensures the keyword is not a mistaken word and created a more accurate mechanism.

The tags extracted is used in order to query the tweets dataset and compile a set of tweets that might be relevant to TV broadcasts. However, by using the whole collection of tags as a mean to search Twitter could not guarantee that all the tweets relevant to a certain TV-series are returned. Some twitter users may compiled tweets by using words which are synonyms to the the words that had been extracted as tags. The solution to this is to enrich the tag collection by querying WORDNET [32] for synonyms. However, the company is more concerned about the efficiency of this system and finding synonyms would add complexity to the tag extraction mechanism and that is the reason for not implementing programmatically this idea.

3.4 Evaluation

It should be made clear that this research does not only concentrate on extracting tags relevant to the corresponding TV subtitles because this would be expected at some level. Hence, measuring the correlation between subtitles and the extracted tags is not the only desired aim. The major aim of the evaluation is to measure the relation between the tags that have been extracted and the corresponding episode of TV-series or Movies. To make it clear, the aim of the whole procedure is to produce keywords that Twitter users likely use when compiling tweets for the certain TV-series. As a result when these keywords are used as search terms tweets relevant to the corresponding TV-Series could be found.

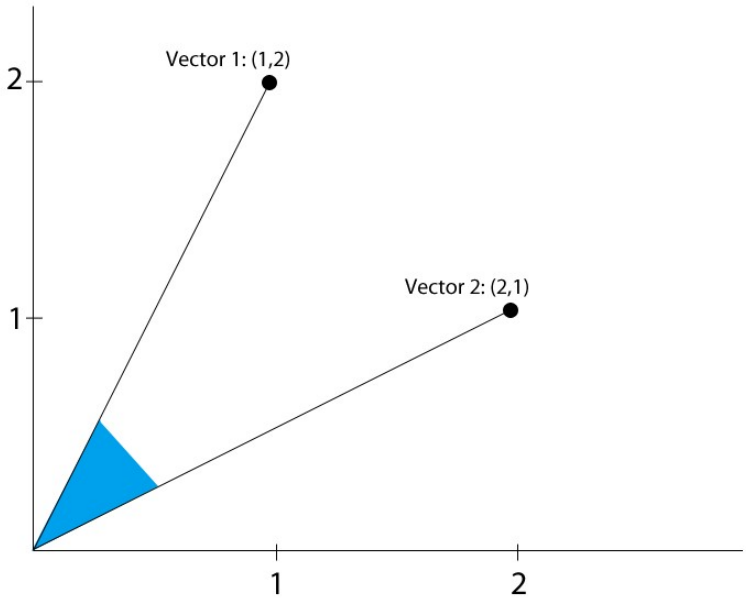
Thus, instead of comparing tags with subtitles, the evaluation method that had been examined reveals the correlation between tags and the description of the corresponding broadcast's episode crawled by IMDB. The measure which had been used for the evaluation of the accuracy of the tag extraction system is called cosine similarity.

3.4.1 Cosine Similarity

Cosine similarity[30] is a measure used frequently in data mining and indicates the similarity between two texts. The texts are represented as vectors in a N-Dimensional space where N is the number of the unique words that exist in both texts. Explaining it further, the cosine of the angle that is created between these vectors could reveal their similarity. if the vectors point at the same direction it means that the represent texts are correlated. As a straightforward example, assume that the following texts should be compared:

1) white white cat
2) white cat cat

The x-axis represents the frequency of the word cat and the y-axis represents the frequency of the word white in the texts.



The cosine of the angle that created between the two vectors reveals the similarity of the texts. Mathematically, the cosine similarity is defined by the Euclidean dot product formula:

$$a \cdot b = \|a\| \|b\| cos\theta$$

Assume that $V1$ is the vector derived by the summary of the TV-series text and $V2$ is the vector that derived from the extracted tags. Hence, the cosine similarity formula is:

$$similarity = cos\theta = \frac{V1 \cdot V2}{\|V1\| \|V2\|}$$

where:

$$V1 \cdot V2 = \sum_{i=1}^{n} V1_i \cdot V2_i$$

$$\|V1\| \|V2\| = \sqrt{\sum_{i=1}^{n} (V1_i)^2} + \sqrt{\sum_{i=1}^{n} (V2_i)^2}$$

In order to evaluate our tag extraction system, three TV-series are used: 1)Doctors, 2)EastEnders, 3)Call The midwife. For these TV-series, tags from its corresponding subtitles had been extracted. Afterwards the descriptions of the corresponding episodes had been crawled from IMDB. Finally, the tags and the corresponding description had been converted to vectors. These vectors were compared by using the cosine similarity measure in order to reveal their similarity.

The results are presented in the following table:

TV-Series	Cosine Similarity	Similarity (Approximately)
Doctors	0.9038741156756381	90%
EastEnders	0.6686478498357316	67%
Call the midwife	0.8798446823111474	88%

Table 3: Similarity between the extracted tags and the corresponding description for the examined TV-series

The above similarities validate the fact that the extracted tags are correlated with the corresponding TV-series. The results could be improved further by adding in the tags list synonyms of the extracted tags or knowledge related to the corresponding broadcast. However, this would raise the complexity of the system and would reduce the time efficiency that this system achieves.

3.4.2 Time efficiency

Another characteristic which should be examined is the total amount of time that this system demands in order to fulfil its aim for an efficient tag extraction system. This is very crucial because SecondSync platform must export results very fast in order to analyse TV-information in real-time. As a result the tag extraction system should extract keywords for a given TV-series in less than 30 seconds (this threshold was given by the company). The following table presents the efficiency that the current extraction mechanism achieves for each TV-series:

TV-Series	Time (in seconds)
Doctors	6.0
EastEnders	3.0
Call the midwife	3.0

Table 4: Time needed for extracting tags from each TV-series

The times presented include the total time for accomplishing all of the following tasks:

- Dataset Corrections
- POS-Tagging
- Tokenization of the whole collection of the subtitles.
- Tokenization of the given TV-series subtitles.
- Tag Extraction

It is observed that the system does not exceed the threshold that had been set but also it achieves the extraction in only a small period of time. However, SecondSync platform stores larger subtitle datasets and as a result more time is needed in order to accomplish the tokenization procedure. This could be resolved by doing the tokenization (of the whole collection of the subtitles) only once and afterwards store the tokens in a database instance along with their TF scores.

3.5 Chapter Summary

This chapter presented a tag extraction system that suits TV subtitles. The aim of this system is to produce keywords that are being used as search terms for querying tweets datasets. The returned results contain several tweets which might be relevant to TV-Series. A robust filtering system is responsible for identifying the desired tweets. This mechanism is using an improved model that takes into consideration the produced tags in order to achieve a better classification. As a result, it is clear that the described system is directly connected with the filtering system that is described in the next chapter. Finally, the evaluation methods revealed that the system is successfully exporting terms which have high similarity with the corresponding TV-broadcasts.

4. Tweets Filtering

SecondSync is a platform manipulating huge amounts of tweets datasets. Among these, some information is useless and some information is useful in order to produce the TV-analytics. Hence, some kind of filtering should be done in order for the system to maintain only the useful tweets.

In more detail, tags extracted from the previous task are used as search terms* for querying large tweet datasets. Some of the returned tweets are irrelevant to TV broadcasts and some of them are relevant. This kind of separation is a challenging task in computer science and usually could be resolved by using Machine Learning Techniques. Specifically ML methods, based on an appropriate model, are applied in order to create two different probabilistic classes of tweets (Relevant/Irrelevant Tweets). Subsequently, each tweet in the returned results is firstly filtered and then assigned to one of the probabilistic classes.

The whole procedure is called information filtering and there are a lot of studies using social networks and such techniques to identify information relevant to earthquakes [1], traffic [2], epidemics [3], Elections [4], News [5], TV [6]. The present research is focused on studies and tries to improve the filtering system which was introduced in [6].

4.1 Machine Learning Fundamental Notions

ML is a sector of Artificial Intelligence where given an assumption (trained data), an outcome could be predicted. The algorithm that predicts the outcome is called classifier and is responsible for separating outcomes (relevant or irrelevant tweets in that case) into categories (classes). This task is called classification and usually is using a model in order to calculate the probability of an outcome to happen or not. The outcome with the bigger probability reveals the class of the outcome.

In this study, a fair amount of tweets are labeled as relevant or irrelevant to TV broadcasts. These tweets form the training set of the filtering system suggested by this study. These tweets have some features used in our model. Such features could be the following: tweets relevant to TV broadcasts frequently contain TV terms or actors names. Evaluating these characteristics our system could predict the class (relevant or irrelevant) of a completely random tweet, based on its own characteristics.

In ML there are a lot of classifiers that have both advantages and disadvantages. This study is presenting a ML model based on the naïve Bayes classifier. This choice had been made because Bayes classifier possesses a great advantage: It requires only a small amount of trained tweets in order to produce outcomes accurately. This is useful because the provided tweets dataset is limited due to copyright restrictions. Usually, a training set consists of million tweets which is not the case for this research. This is the reason why the naïve Bayes is the perfect solution for this project research.

* The company uses also search terms such as the title of the broadcast or the channel in order to retrieve more results.

4.2 Bayes Theorem

The intuition behind Bayes theorem is based on the fact that when given a knowledge A that could be observed, the outcome of an event B could be predicted. The formal definition of this is given below:

$$P(B|A)=\frac{P(A|B)\cdot P(B)}{P(A)}$$

where $P(B)$ is called a-priori and defines the probability of B to be happened without any prior knowledge. Furthermore, the element $P(B|A)$ is called posteriori and defines the probability of the event B to be happened when knowing the probability A . This theorem is the core of the naïve Bayes machine learning algorithm.

4.3 Naïve Bayes

This algorithm[31] is a binary classifier and is the core of our filtering phase. It is called binary because decides if a random tweet is or is not relevant to TV broadcasts. As already mentioned before, its big asset is the ability to be trained only with a small corpus of tweets. Our option of choosing this classifier is verified by [5] where Sankaranarayanan, Samet, lieberman and Sperling had used the same classifier in order to filter news messages crawled from Twitter.

Naïve Bayes selects the best classification (relevant or irrelevant) for a random tweet based on the state of the features $f_1, f_2, \dots, f_n$. The classification bases on Bayes theorem and is defined as follows:

$$classification(tweet_k)=argmax\ P(C=c|f_1,\dots,f_i)$$

$$argmax\ \frac{P(f_1,\dots,f_i|C=c)\cdot P(C=c)}{P(f_1,\dots,f_i)} =$$

$$= \quad argmax\ P(f_1,\dots,f_i|C=c)\cdot P(C=c)$$

The classifier is called naïve because independency between the features f is assumed. Hence,

$$classification(tweet_k)=argmax\ p(C=c)\prod_{i=1}^{n} P(F_i=f_i|C=c) \qquad (1)$$

where c represents the class (relevant/irrelevant) and $p(C=c)$ the probability of this class. $p(F_i=f_i|C=c)$ Is the conditional probability denoting how likely a tweet with a certain feature F_i belongs to the class c . The conditional probability could be calculated by the bayes theorem (using m-estimates), as follows:

$$P(F_i = f_i | C = c) = \frac{n_c + m \cdot p}{n + m} \quad (2)$$

where F_i is the feature and f_i is the corresponding value. In addition, n_c is the number of tweets in the training set for which $C = c$ and $F_i = f_i$, p is the a-priori estimation for $P(C = c | F_i = f_i)$, n is the number of tweets in the training set for which $C = c$ and m is the equivalent sample size. In addition, F_i is the feature and f_i is the corresponding value. A general view on how this classifier is used is explained in the following paragraph:

Assume that there is a training Set with various tweets. Some of them are considered as relevant to TV broadcasts and some of them as irrelevant. As a result, the class of each tweet could be either “relevant” or “irrelevant”. The class is assigned manually to each tweet of the training set. In addition, each tweet has some specific features and the important is that some tweets which have some certain features are more likely to belong in one of the classes. This fact helps to assign a random tweet in a certain class and is done by the naïve classifier which helps to identify messages relevant to TV broadcasts or not.

When a new tweet is inserted in the system it is checked for the number of predefined features. Afterwards for each feature, the probability (2) is calculated for $C = relevant$. The same methodology is followed for $C = irrelevant$. The classification with the biggest probability (calculated from (1)) reveals the class of the random tweet.

One major issue which has not been discussed yet is what are the exact characteristics that may reveal the class of a random tweet. Below are presented the existing methods and characteristics known for classifying tweets relevant to TV broadcasts [6].

4.4 Existing methods

Dan, Feng and Davidson were introduced in [6] a classifying system identifying micro-blogging messages relevant to TV-broadcasts. This study represents a system with many features, two training sets and two classifiers. As long as our system is based on this study, it is worth shortly presenting it despite the fact that our research is using only one classifier and some additional specialized features improving the whole procedure.

First Classification:

The first classification system is using a training set that consists of 1000 manually labelled tweets (labelled as relevant or irrelevant to TV broadcasts). The tweets are labelled from workers hired from Amazon Mechanical Turk*. This task is using the Rotation Forest classifier[33]. Moreover, as described above, each tweet had been checked for some features. When a feature is presented in a certain tweet, it is set to true, otherwise it is set to false.

The features used for this classification are the following:

* This is a marketplace for finding workers for artificial intelligence tasks such as training set compiling.

- ***Related Terms***
  This feature checks if a message contains expression that are related to television. In more detail a table with various expressions considered as TV relevant was compiled. Some examples of such phases are: “watching”, “tv-series”, “season”, “the actors in” etc.

- ***Channel Terms***
  A lot of tweets related to broadcasts contain the channel in which they are aired. Specifically, if a tweet contains tokens such as “bbc”, “cnn” or other channel related information, the current feature is set to true.

- ***Episode Terms***
  This feature includes checks for TV series related messages. Some users tend to accompany their tweets with episode and season abbreviations instead of writing expressions such as “season 01, episode 01”. Examples of such abbreviations are: “S01E03”, “S1E1” etc. These texts are commonly used in Twitter mainly because of the fact that micro-blogging messages are limited to 140 characters. In order to “save” space it is convenient for the users to use expressions similar to these presented above. Such text patterns could be “grabbed” by using special regular expressions.

- ***Syntax Rules***
  All the above characteristics are not enough in order to identify TV related messages successfully. Thus, some more advanced features were introduced. One of them is the syntax rules feature. A list of syntax rules commonly used in tweets relevant to TB broadcasts was constructed. Some rules in this list are presented in the table below:

<TEXT> <ACTOR NAME> in the episode	<TEXT> watching <BROADCAST TITLE>
Episode of <BROADCAST TITLE>	<CHARACTER NAME> was cool

  Where <TEXT> is a tag that indicates general text, <BROADCAST TITLE> indicates the title of a broadcast and the <ACTOR/CHARACTER NAME> tag describes words of the message that represents the name of actors/characters.

- ***Title Capitalization***
  According to [6], Twitter users tend to capitalize the titles of the broadcasts. A feature is created for this habit.

- ***Multiple Titles***
  When multiple broadcast titles appears in a text it is considered as feature as well.

- ***Actors Cosine***
  The authors of [6] created a crawling system that finds additional information for each broadcast in online sources. These sources include Wikipedia and TV.com. The actors names are collected and then then are compared to the tweet message by using the cosine similarity rule.

- ***Characters Cosine***
  It is as the actors cosine feature but the character names are checked.

- ***Side Information Cosine***
  This features is using the cosine similarity rule in order to compare the knowledge extracted from the broadcast's wikipedia page.

Second Classification:

The training set of the second classifier consists of tweets classified as relevant from the first classifier. This set is increased dynamically each time the first classification takes place. The underlying classifier selected for this task is J48 which is based on C4.5[34]. The features used for this classification are the following:

- ***Positive Syntax***
  This feature is derived from the characteristic named “Syntax Rules”. It searches for positive syntax in a message. For example a positive type of syntax is an expression such as “episode of house”.

- ***Negative Syntax***
  This feature is the opposite of the feature described above. It contains syntax which is not referred to TV broadcasts. Such syntax could be the following: “the white house”.

- ***HashTag score***
  All the messages that had been classified as relevant to TV broadcasts from the first classifier, may contain some hashtags. Commonly occurring hashtags are captured and added in a list. A tweet could be checked for these hashtags in order to be decided if this feature should be set as true or not.

- ***User score***
  Using the first classification, users publishing more often TV related tweets are captured. A random tweet could be checked whether it is published from these users or not. If it is, then this feature is set as true or false.

- ***Rush time***
  This feature checks the habit of twitter users to publish TV-related tweets during the period of time that a certain broadcast is on air. During this period of time, the number of tweets discussing about the broadcast is bigger than any other moment. As a result, when the filtering of a certain tweet takes place, it is checked for where it is published during this interval of time. The key point is that the system does not know the exact airing time of the broadcast but it makes an assumption about it with the above methodology.

4.5 Improvements

The main barrier for using the system described above is that this model cannot be embedded in the SecondSync platform. The reason is that it is very complex and it would not be able to filter information both in real-time and time efficiently. This fact arises the need for a simpler system to be constructed. The new system should be time

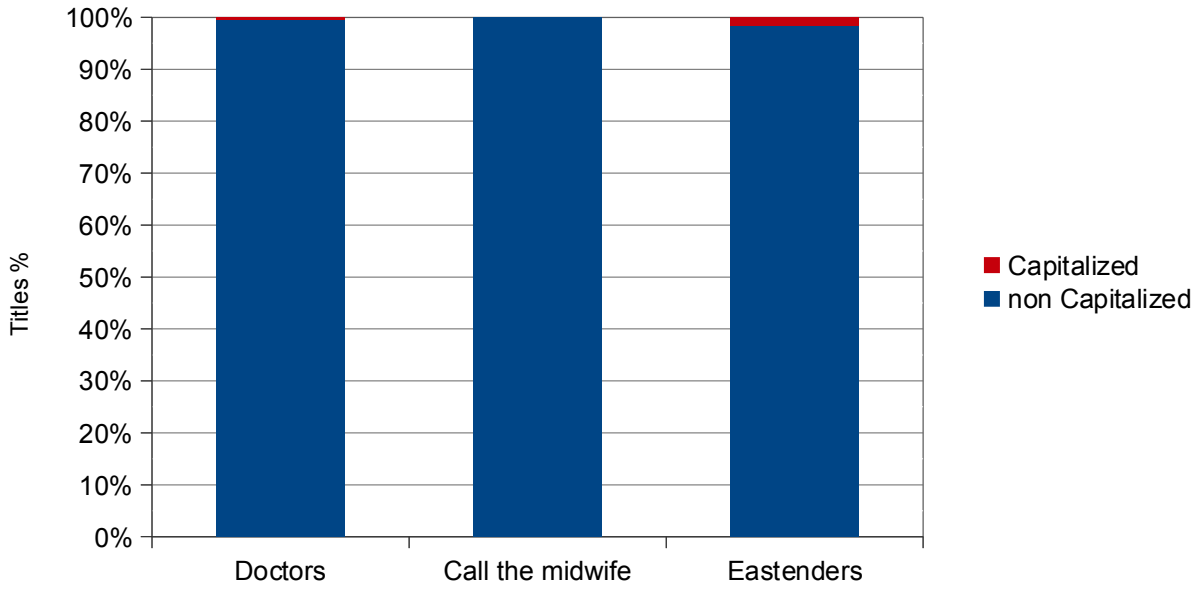
efficient and also able to identify tweets relevant to TV broadcasts with the success that the existing methods do. In fact, the filtering system that this research is suggesting is slightly improved compared to the system described in [6]. This is feasible because the datasets provided by the SecondSync company can help this task to be accomplished. The simplification of the existing system begins with the capitalization feature.

1) Initially, a big collection of tweets was compiled by searching the twitter. Specifically, we built a crawling system using the Twitter search api for querying the social network for messages containing the titles of three BBC's TV-series: Doctors, EastEnders and Call the midwife. A detailed analysis was carried out on this dataset and was found that the capitalization feature does not produce a significantly better classification. The results are summarized in the following table:

TV-series	Tweets with capitalized title	Tweets without capitalized title
Doctors	0.5%	99.5%
Call the midwife	0%	100%
EastEnders	2%	98%

Table 5: Results that reveal the percentage of TV-series related Tweets that contain capitalized titles

The same results are represented visually in the following graph:



Graph 6: A graphical representation of table 4

The results presented above lead to the realization that the current feature could be excluded from our model as only a tiny amount of titles were capitalized and this feature would bring additional complexity to the system which is not desired. Actually, the number of the titles that was found to be capitalized was negligible and as a result there is no reason to include this feature in a TV-Series classification.

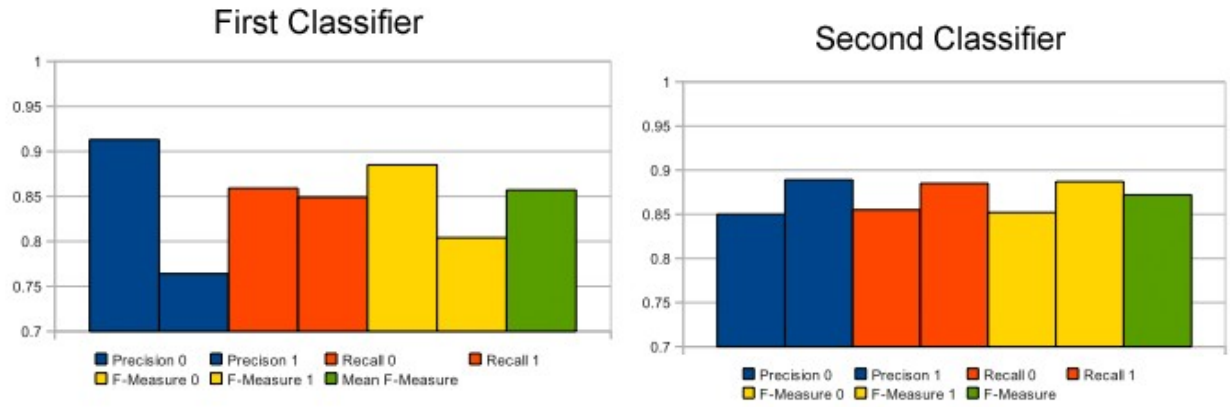
2) As far as the multiple titles feature is concerned, it could be omitted because SecondSync platform is producing analytics only for individual series, so the filtering task should be made for tweets that concern specific series.

3) Moreover, the preprocessing that had to be made (crawl information from different online sources) in order to produce the actors/characters cosine could be replaced by a system that extracts information directly from IMDB. The content contained in this website is more accurate and easily accessible through various apis.

4) In addition, the side information cosine could be replaced by a feature based on the tag extraction phase. Instead of searching information on the internet, as the side information feature demands, the collection of tags produced from subtitles could be used to check this similarity. This reduces the complexity of the filtering system as a tag collection is already available from the previous phase (tag extraction phase).

5) Last but not least, the rush time feature is very clever and is able to make good guesses about the time that a broadcast is on air. However, some calculations need to be done regarding the amount of tweets published during the different periods of time. By using the TV-listings dataset, the additional calculations are avoided as all the airing times for each broadcast are listed.

These were some thoughts about how the existing method could be simplified and improved simultaneously. Apart from that, there are some difficulties appeared in the above filtering method, such as error propagation. This is caused by the fact that the training set of the second classifier consists of tweets classified as relevant from the first classifier. As a result the second training set is not manually compiled and contains false positives. According to [6], the evaluation of the system is presented to the following graph:



Graph 7: These graphs are the evaluation of the system described in [6]. With 0 are labelled the tweets that are irrelevant to TV broadcasts and the 1 label indicates tweets that are relevant to TV broadcasts.

As the reader may observe, the precision of the first classifier for identifying correctly the negative results is 91%. This fact means that the remaining 9% consists of false positives which will be contained in the second training set as well. For the second classifier, the precision for identifying correctly the negative results is decreased to 85%. As a result the number of the false positives have been raised.

The next chapter presents our proposed filtering model which identifies tweets relevant to TV broadcasts and reduces the amount of the false positives that may be produced by

existing methods.

4.6 Suggested Method

In this chapter the filtering method that this research suggests is presented. Our system is based on naïve Bayes classifier. This choice had been made by the fact that the datasets that we had in our possession were limited and this classifier could be trained with small sets and produce efficient classification. Our classifier accepts as input a unique id that represents a certain TV-series and one or more tweets. Finally, it is decided if the tweet is relevant to that system or not.

Assigning an identification number to a tweet is a simple procedure. In order to produce analytics for a certain TV series, the SecondSync platform would search for tweets by using the tags extracted from the tag extraction system. As a result, all the results returned from this search task are linked with the id of a certain TV series.

As far as the features created for our model are concerned are described below:

- ***Actors Feature***
  This feature is similar to the one above but concerns the character names that the actors have in a TV-series. An automated crawling system was built for retrieving the real actor names from the IMDB database. Each TV-series episode has a unique id in IMDB so the list with the actor names is every time up-to-date.

- ***Characters Feature***
  Usually, tweets relevant to TV broadcasts contain character names. This realization could be captured by this feature. In order to use this feature, a table with the names of the characters of the examinable series was constructed. The table compilation was made from information gathered from IMDB.

- ***Network Terms Feature***
  When a tweet contains TV-channel terms, it is likely relevant to TV broadcasts. This feature represents this intuition. A list with all the UK channels was constructed in order for this feature to be effective.

- ***Positive Rules Feature***
  As mentioned in [6], There are some common syntaxes used in tweets containing information for TV broadcasts. A list with some phrases that could be met in such tweets was compiled. For instance:

  watching <title>

  <title> was awesome

  now in <channel>

  <title> is starting

  <character> is so cute

  Where the title of the broadcast replaces the <title> tag, the channel name replaces the <channel> tag and <character> could be either the real name of an actor participating in the TV-series or the corresponding character's name. This feature is using the lists compiled for the actors feature and the characters feature respectively

- ***TV Terms Feature***
  This feature captures the phenomenon where tweets relevant to TV broadcasts contain terms directly connected to television. Examples of common words are:

watch	television	episode	tv	series
channel	movie	actor	scene	casting

- ***Time feature***
  As seen from the SecondSync analytics, there is a huge raise of messages for a certain broadcast when this is on air. This feature is derived from this phenomenon.

- ***Tags Cosine Feature***
  This feature is used in order to capture any similarity between a certain TV-series episode and a tweets content. The intuition here is to compare the content of the tweet and the description of a certain episode. The episode is described accurately by the tags extracted from our tag extraction system. These tags had been stored in a list. In addition, by the fact that the Bayes classification demands as more independency as possible between the features, any hashtags that may be generated by the title or the channel of the current broadcast are added to the tags list.
  As a result, the comparison of the tags list and the content of the tweet will reveal if there is any similarity. This is done by using the cosine similarity rule. When the similarity exceeds a certain threshold, this feature is set to 1.

- ***TF-IDF score feature***
  A very challenging task is to identify micro-blogging messages relevant to TV broadcasts with controversial titles. For instance, the TV series title “EastEnders” is unique, so tweet messages containing this title are relevant to this broadcast with high probability. On the other hand, titles such as “doctors” or “monk” could be contained in Tweets that are not relevant to any broadcast. Examining the title of the broadcast if it is used frequently or not could solve this problem. Furthermore, providing that the TF-IDF score for each word is available in a hashtable from the previous task that we have been implemented, it is easy to decide about this feature in constant time.

Apart from these features, one more feature could be used in order to capture twitter profiles that publish only TV content (TV channel profiles, Actor profiles etc.). An efficient way to check such profiles is to parse the info in the profile description. The given datasets did not contain this information and as a result this feature was not implemented. However, it is feasible and the SecondSync platform has the ability to search for profile information when captures tweets.

It should be mentioned that the above combination of features was the one producing the best result. Some other features were tested but they did not produce the desired results as the above combination did. However, besides finding the best combination of features, this research applied a quick preprocessing in the tweets datasets in order to perfect the system.

In more detail, before assigning a tweet to a certain class (relevant or irrelevant), a quick preprocessing should be made in order to assure that the analysis of the features will not be affected by external factors. This task includes the tokenization of each current tweet and then checks for common errors. A common error that may occur is the following: when somebody replying to a message published from another user of twitter he/she uses a token as this: @<username>. If this username is the same as the name of an actor/character then the corresponding feature is set to true which is not correct. As a result, not taking such tokens into account resolves this problem. Furthermore, tokenization of the tweet enables us to create the suitable vectors and finding the similarity needed for the tags cosine feature easily. Finally, in the case where the current tweet is identified as relevant to a TV broadcast, the tokenization allows us to store any hashtags (if any) that is contained in the message and use them later for further analysis.

The aim of the developed system was to simplify the existing methods and produce the same efficiency and reduce the amount of false positives. Theoretically, the false positives could be reduced because the fact that some TV series have controversial titles (TF-IDF feature) had been taken into account. Indeed, the classification system has smaller complexity and produces less amount of false positives compared to the system that the company is using. The results are presented in the following chapter in detail.

4.7 Evaluation

The evaluation was done by filtering tweets for the following TV series:

- Doctors
- EastEnders
- Call the midwife

In order to produce a more detailed analysis for the filtering system four types of outcomes had been considered for each tweet:

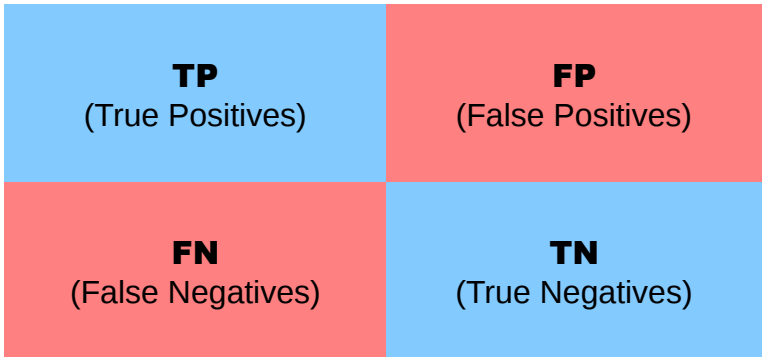


FIGURE 6: This figure represents the four different outcomes for each tweet

The sets coloured with red represent mistaken classification and the sets coloured with blue represent correct classification. Both FP and FN are undesired outcomes but the current research is more concerned about FP.

- **True Positives**
  TP represent correct classification for tweets relevant to TV broadcasts. In more detail, this set includes tweets that theoretically belong to the class “relevant” and

indeed our classifier identified them correctly.

- **False Positives**
  FP represent mistaken classification for tweets irrelevant to TV broadcasts. In more detail, this set includes tweets that theoretically belong to the class “irrelevant” but our classifier identified them as “relevant”.

- **True Negative**
  TN represent correct classification for tweets irrelevant to TV broadcasts. In more detail, this set includes tweets that theoretically belong to the class “irrelevant” and indeed our classifier identified them correctly.

- **False Negatives**
  FN represent mistaken classification for tweets relevant to TV broadcasts. In more detail, this set includes tweets that theoretically belong to the class “relevant” but our classifier identified them as “irrelevant”.

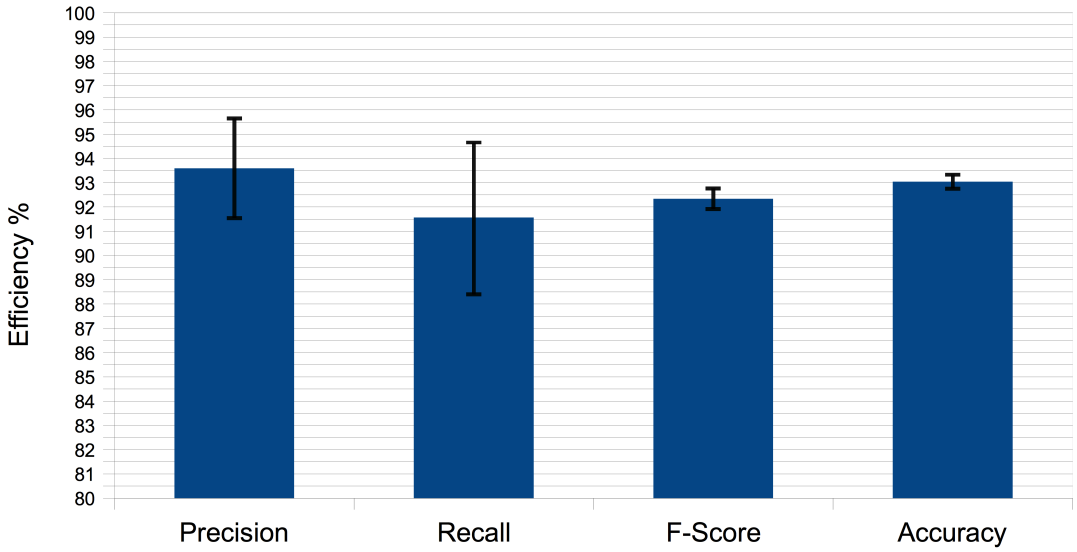
The above separation enables us to use commonly known measures that are used in order to evaluate ML techniques. Specifically, the efficiency of this technique is shown by measuring the accuracy, precision, recall and f-score of the filtering system. The formula for each measure and what represents is defined below:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

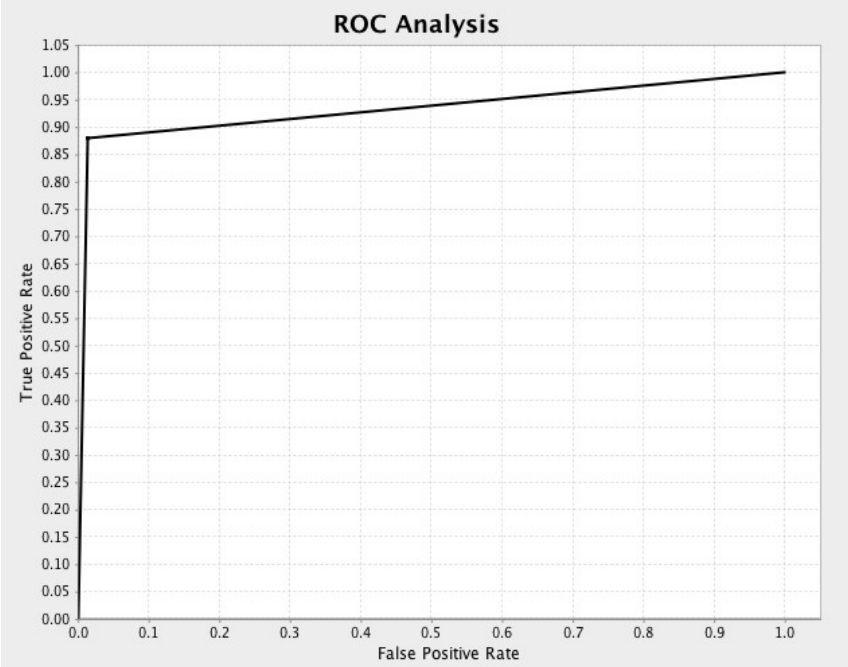
The accuracy measure illustrates the correctness of the classification system. Precision indicates how likely is for the system to do correct predictions about relevant tweets. Recall reveals the ability of the system to choose elements of the relevant or irrelevant class from a random set of tweets. In order to evaluate the system, three different sample sets were compiled consisting of 60-150 messages each. The graph which is presented below summarises the efficiency of the filtering system for each one of the above measures:



Graph 8: A graphical representation of the developed classifier's efficiency

Besides the fact that the above graphs present a successful classification. It should be noted that the tests had been made for three TV-series only because the tweets and subtitles datasets were limited (due to copyright reasons).

Apart from identifying TV related tweets correctly, another fact that should be considered is the efficiency of the system for the produced false positives. The most suitable method that could be measured is the ability of the system in producing only a small amount of false positives and is called ROC analysis. The ROC curve for our system is presented below:



Graph 8: ROC curve of the developed classifier

The upper left corner of this graph illustrates a perfect classification for the false positives. As the reader may observe, the ROC graph of our filtering system increases fast and reaches a point which is very close to the point representing the classification without false positives. This fact indicates that our filtering system is very efficient and produces only a small amount of false positives.

4.8 Chapter Summary

This chapter presented a filtering system which efficiently identifies TV related twitter traffic. This was feasible by building a bayesian classification model based on TV characteristics. Furthermore, this model was improved The lowest precision and recall that the developed classifier achieved are 91,5% and 88,4% respectively. As far as the the false positives are concerned, the ROC curve showed that the perfect point (perfect classification as far as the false positives are concerned) was approached. Especially regarding false positives, SecondSync filtering platform was producing poor results for TV-series with controversial titles. This fact was revealed when two different filtered sets were compared. These sets constitute the results of this research's classification system and SecondSync's classifier respectively. For the “Doctors” TV-series the filtering system that this study recommends reduced the false positives by 76,4%. However, due to copyright reasons, the efficiency of the current task evaluated by using a limited set of tweets and TV-series.

6. CONCLUSION

6.1 Critical Evaluation

As described in the beginning of this study, the current project is exploring methods for constructing a tag extraction system that suits subtitles. Apart from that, a major aim was to construct an improved and efficient filtering system which identifies tweets relevant to TV broadcasts. The most important concern about the desired system is the time efficiency and the filtering accuracy that it achieves. Existing methods not only produce poor results as far as the false positives are concerned but also are too perplexed to be used in a system applying real-time analysis. In order to achieve the goals that this research had set, a detailed exploration had be done among various methods and was used for extracting tags and filtering information. This exploration led the research to adopt the most suitable methods for achieving the desired aims. Specifically, it is constructed a tag extraction method that suits subtitles. This method was based on [29]. However, some improvements and modifications had been made in order to use this method to TV subtitles. Furthermore, it is constructed a filtering system which is time efficient but also accurate, as far as the false positives are concerned. The appropriate classification model was inspired by the research presented in [6]. According to the aims and the objectives that had been set at the beginning of this project, the developed system is divided in two parts that are analytically described below.

The first part is about finding methods for extracting accurate keywords that describe certain TV-series. These keywords are used as search terms for querying big tweet datasets. The search result include tweets that might be relevant to TV broadcasts. The research reached to the conclusion that the most efficient scientific method for accomplish this task is by applying NLP methods in the subtitles of the desired TV-series. This procedure produces tags that are directly connected to the corresponding broadcast. In addition, as long as the subtitle datasets is being generated by automated Text-To-Speech Technologies, it is anticipated that they are going to contain a lot of misspelled words. This fact is an important barrier In order to construct an efficient tag extraction system. As a result correction procedures applied to the given dataset in order to be successful. Finally, The developed system was evaluated by comparing TV-series episodes and their description (crawled on IMDB). The comparison made by using cosine similarity. Furthermore, it is important to be mentioned that this task is using a statistical approach to solve the tag extraction problem. This approach had been selected as it outperforms similar Machine Learning methods.

The second phase illustrates a filtering system that identifies tweets relevant to TV broadcasts. Specifically, this part is about filtering the results that were returned when querying the tweets dataset with the extracted tags from the first phase. The developed filtering system was based on naïve Bayes classification and some of the features that our classifier is using were inspired from the work of Dan, Feng and Davidson [6]. However, our filtering system is more simplified fact that reduces its complexity. This element is very important as long as the developed system should cope with the efficiency that SecondSync platform demands in order to produce real-time TV analytics. Apart from the features that were inspired by [6], some new features had been introduced

derived from the tag extraction system. The main aim of these features is to capture tweets relevant to TV broadcasts with controversial titles. Our classification system achieved very good accuracy thus it identified TV related activity successfully. Lastly, the evaluation revealed that the false positives amount was decreased by 76,4% compared to the classification mechanism that SecondSync is already using.

In conclusion, the current research fulfilled its aims and objectives by suggesting and developing the above methodologies. The evaluation made by using datasets from three different TV-series. Despite the fact that the sample set was representative, the final results could be more robust, if further evaluate them by using bigger datasets. The used datasets were limited due to copyright reasons.

6.2 Future Work

As future work, the existing system could be tested with larger sample sets in order to yield more robust results that validate the efficiency of the filtering system and were presented in this report. Furthermore, some additional filtering techniques could be done in order to identify tweets relevant to various types of broadcasts such as news, sports news, documentaries and talk shows.

As far as the news broadcasts are concerned, a basic research revealed that there are some methodologies based on clustering that could be valuable for future research for this task. As mentioned in the background chapter, the news filtering is very challenging because of the fact that news are generated continuously. Furthermore, it is not easy to extract distinct features from content changing continuously. However, taking into consideration various news websites or twitter profiles that publish messages related to news,the construction of a dynamic set of news information is possible. This set represents various news that are happening right now. Separating this information into clusters gives a mechanism that could be used in order to identify distinct news. The first challenge is to match the different clusters to concepts (n-grams of keywords) appeared in the subtitles of broadcasts which are relevant to news. Once it is accomplished, the second challenge is to use the dynamic set as complementary training set to train a classifier for filtering tweets returned when querying the tweets dataset with the concepts extracted from the subtitles. The big asset is that the dynamic set is continuously updated with the most recent news. Finally, the complexity of the system could be reduced by deleting the clusters that haven't been updated for a certain period of time.

BIBLIOGRAPHY

[1] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes Twitter users”, 2010, p. 851.

[2] N. Wanichayapong, W. Pruthipunyaskul, W. Pattara-Atikom, and P. Chaovalit, “Social-based traffic information extraction and classification”, 2011, pp. 107-112.

[3] V. Lampos, T. D. Bie, and N. Cristianini, “Flu detector: tracking epidemics on twitter,” 2010.

[4] S. Sudhahar, T. Lansdall-Welfare, I. N. Flaounas, and N. Cristianini, “ElectionWatch: Detecting Patterns in News Coverage of US Elections”, 2012, p. pp.82-86.

[5] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, “TwitterStand,” 2009, p. 42.

[6] O. Dan, J. Feng, and B. Davison, “Filtering microblogging messages for social tv”, 2011, p. 197.

[7] A. Pak, P. Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining”, 2010.

[8] J. J. Webster and C. Kit, “Tokenization as the initial phase in NLP”, 1992, vol. 4, p. 1106.

[9] Voutilainen, A., Heikkil, J., Anttila, A. “Constraint grammar of English: A performance-oriented introduction”, 1992.

[10] F. Karlsson, A. Voutilainen, J. Heikkil, and A. Anttila, “Constraint Grammar”.

[11] L. Rabiner and B. Juang, “An introduction to hidden Markov models”, *Magazine*, vol. 3, no. 1, pp. 4-16, 1986.

[12] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification,” *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3-26, Jan. 2007.

[13] R. Krovetz, P. Deane, and N. Madnani, “The web is not a person, Berners-Lee is not an organization, and African-Americans are not locations: an

analysis of the performance of named-entity recognition," 2011, pp. 57-64.

[14] M. Marrero, S. Sanchez-Cuadrado, J. Morato, and Y. Andreadakis, "Evaluation of Named Entity Extraction Systems," 2009, pp. 47-58.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software", *{ACM} {SIGKDD} Explorations Newsletter*, vol. 11, no. 1, p. 10, Nov. 2009.

[16] F. Eibe, M. Hall, G. Holmes, R. Kirkby, and I. H. Witten, "WEKA: A Machine Learning Workbench for Data Mining".

[17] B. Haberd et al., "Towards Tokenization Evaluation"

[18] J. J. Webster and C. Kit, "Tokenization as the initial phase in NLP", 1992, vol. 4, p. 1106.

[19]D. D. Palmer and M. A. Hears, "Adaptive Multilingual Sentence Boundary Disambiguation", vol. 23, pp. 241-267, 1997.

[20] S. Kotsiantis, D. Kanellopoulos, and P. Pintellas, "Data Preprocessing for Supervised Learning", 2006.

[21] P. Turney, Extractor http://www.extractor.com

[22] D'avanzo, E., Magnini, B., Vallin, A., Keyphrase Extraction for Summarization Purposes: The Lake System at DUC-2004

[23] E. D'Avanzo, B. Bagnini, and A. Vallin, "Keyphrase Extraction for Summarization Purposes: The LAKE System at DUC-2004", 2004.

[24] S. R. El-Beltagy and A. Rafea, "KP-Miner: A keyphrase extraction system for English and Arabic documents", *Information Systems*, vol. 34, no. 1, pp. 132-144, Mar. 2009.

[25] CâMpeanu, K. Salomaa, and S. Yu, "A FORMAL STUDY OF PRACTICAL REGULAR EXPRESSIONS", *International Journal of Foundations of Computer Science*, vol. 14, no. 06, pp. 1007-1018, Dec. 2003.

[26] B. Carle and P. Narendran, "On Extended Regular Expressions", in *Language and Automata Theory and Applications*, vol. 5457, D. Hutchison et al., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 279-289.

[27] G. A. Miller and C. Fellbaum, "WordNet then and now", *Language*

Resources and Evaluation, vol. 41, no. 2, pp. 209-214, Oct. 2007.

[28] S. Robertson, “Understanding Inverse Document Frequency: on Theoretical Arguments for IDF”, 2008, pp. 503-520.

[29] M. Dostal and K. Jezek, “Automatic Keyphrase Extraction based on NLP and Statistical Methods”, 2011, pp. 140-145.

[30] S. Tata and J. M. Patel, “Estimating the selectivity of tf-idf based cosine similarity predicates,” *{ACM} {SIGMOD} Record*, vol. 36, no. 2, pp. 7-12, Jun. 2007.

[31] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[32] G. A. Miller, “Princeton English Lexical database”, 2012, http://wordnet.princeton.edu/

[33] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, “Rotation Forest: A New Classifier Ensemble Method,” *{IEEE} Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619-1630, Oct. 2006.

[34] J. R. Quinlan, *C4.5 : programs for machine learning*. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993.

APPENDIX A: CRITICAL PARTS OF THE DEVELOPED CODE

This section contains only a small but critical part of the code that was developed during the implementation of this project. The full software package was submitted with the report.

TF score:

```
package tagExtractor;

import java.util.ArrayList;
import java.util.HashMap;
import resources.TaggedWord;

public class CalculateTFscore {
	HashMap<String, Double> weightCollection = null;
	public CalculateTFscore() {
		weightCollection = new HashMap<String, Double>(1);
	}
	/**
	 * The calculateWeights function takes as input an ArrayList that contains
	 * tokens and their corresponding types. The calculateWeights function returns
	 * a HashMap<String, Double> where the key is a word and the value is the
	 * weight. In essence the HashMap contains all the words of the ArrayList
	 * but not duplicated.
	 *
	 * @param words
	 * @return
	 */
	public HashMap<String, Double> calculateWeights2(ArrayList<String> words) {
		PosTagging tmpPOStag = new PosTagging();
		final Double initialWeight = 1.0;
		Double tempDouble = null;
		String tmpString = null;
		for (String word : words) {
			tmpString = word;
			if (tmpString.charAt(0) >= 65 && tmpString.charAt(0) <= 90) {
				if (!tmpPOStag.isNameWord(word)) {
					tmpString = word.toLowerCase();
				}
			} else {
				tmpString = word.toLowerCase();
			}
			if (!weightCollection.containsKey(tmpString)) {
				weightCollection.put(tmpString, initialWeight);
			}
			else {
				tempDouble = weightCollection.get(tmpString);
				tempDouble = tempDouble + 1.0;
				weightCollection.put(tmpString, tempDouble);
			}
		}
		return weightCollection;
	}
}
```

IDF Score:

```
package tagExtractor;

import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
```

```
public class CalculateIDFscore {
	private HashMap<String, Double> overallTokens;
	private HashMap<String, Double> tokens;
	private HashMap<String, Double> tokensWithIDF;
	/**
	 * This is the constructor of the class CalculateIDFscore. It takes as input
	 * two hashmaps with the format <String, Double>. The first Hashmap contains
	 * all the tokens in the subtitles collection and the corresponding TF score
	 * the second Hashmap contains the tokens of the specified subtitles which
	 * are linked to the specified Tv series. Moreover it contains the TF score
	 * of tokens of this set.
	 *
	 * @param overallTokens, tokens
	 * @param tokens
	 */
	public CalculateIDFscore(HashMap<String, Double> overallTokens,
			HashMap<String, Double> tokens) {
		this.overallTokens = overallTokens;
		this.tokens = tokens;
		this.tokensWithIDF = new HashMap<String, Double>(tokens.size());
	}
	/**
	 * This function returns the value of the specified key in a HashMap
	 * Collection.
	 *
	 * @param HashMap<String, Double>, map
	 * @param String key
	 * @return Double Value
	 */
	public Double getValueByKey(HashMap<String, Double> map, String key) {
		for (Entry<String, Double> entry : map.entrySet()) {
			if (key.equals(entry.getKey())) {
				return entry.getValue();
			}
		}
		return null;
	}
	private void computeScore() {
		Double idf, proportion = 0.0, overallValue = 0.0;
		for (Map.Entry<String, Double> entry : tokens.entrySet()) {
			idf = 0.0;
			String key = entry.getKey();
			Double value = entry.getValue();
			if (overallTokens.containsKey(key)) {
				try {
					overallValue = getValueByKey(overallTokens, key);
				} catch (Exception e) {
					System.out.println("The function
					CalculateIDFscore did not return a Double!");
				}
			}
			proportion = value / (overallValue);
			idf=proportion;
			tokensWithIDF.put(key, idf);
		}
	}
}
```

Tokenization procedure:

```
package tagExtractor;

import java.util.ArrayList;
import java.util.List;
import resources.Subtitles;
```

```
public class Tokenization {
	private ArrayList<String> tokensCollection;
	public Tokenization() {
		tokensCollection = new ArrayList<String>(1);
	}
	private String eraseInvalidCharacters(String sentence) {
		sentence = sentence.trim();
		sentence = sentence.replaceAll("[/\\!:()*._`,;\\n?\"<>|]+", " ");
		sentence = sentence.replaceAll("(-)+", " ");
		sentence = sentence.replaceAll("( )+", " ");
		sentence = sentence.replaceAll("\\n", " ");
		sentence = sentence.trim();
		return sentence;
	}
	/**
	 * This function is responsible for the tokenization of the subtitle texts.
	 * It takes as input a List of Subtitle objects. and returns an
	 * ArrayList<String> with all the tokens
	 *
	 * @param result
	 */
	public void doTokenization(List<Subtitles> result) {
		String tmpSentence = "";
		for (Subtitles subtitle : result) {
			tmpSentence = subtitle.getText();
			tmpSentence = eraseInvalidCharacters(tmpSentence);
			String[] words = tmpSentence.split(" ");
			for (String str : words) {
				if (!str.contains("'")) {
					tokensCollection.add(str);
				}
			}
		}
	}
	public ArrayList<String> getTokensCollection() {
		return tokensCollection;
	}
}
```

Bayes classification:

```
package classification;

import java.util.ArrayList;
import resources.TrainedTable;

public class Bayes {
	public Bayes() {
	}
	public Double calculateProbability(Boolean stateOfFeature,
			Boolean labelWanted, ArrayList<Boolean> featureList,
			ArrayList<Boolean> labelsList) {
		final int sampleSet = 6;
		// our features are Booleans so the can have two values --> 0/1
		// As a result the a_priori possibility for this to happen is 0.5
		final double a_priori = 0.5;
		int thisFeatureSum = 0;
		int strictSum = 0;

		for (int i = 0; i < featureList.size(); i++) {
			// If feature is set for this tweet
			if (featureList.get(i) == stateOfFeature) {
				// Increment the total number of tweets that have this feature
				// as set
				thisFeatureSum++;
				// if the current tweet is labeled as YES
```

```
                                if (labelsList.get(i) == labelWanted) {
                                        // Increment the total number of tweets that have this
                                        // feature as set nd also are labeled with the desired Label
                                        strictSum++;
                                }
                        }
                }
                double probability = (strictSum + sampleSet * a_priori)
                                / (thisFeatureSum + sampleSet);
                return probability;
        }
        /**
         * This function returns the Bayes probability
         */
        public Double clasify(Boolean actorsFeature, Boolean charactersFeature,
                        Boolean networkFeature, Boolean rulesFeature, Boolean tagsFeature,
                        Boolean tvTermsFeature, Boolean timeFeature, Boolean tfFeature,
                        Boolean label) {

                TrainedTable table = null;
                TrainSet set = new TrainSet();
                table = set.getTrainedTable();

                Double actors_pr, characters_pr, network_pr, rules_pr,
                        tags_pr, tv_pr, time_pr, tf_pr;

                actors_pr = probabilityActors(actorsFeature, label, table);
                characters_pr = probabilityCharacters(charactersFeature, label, table);
                network_pr = probabilityNetwork(networkFeature, label, table);
                rules_pr = probabilityPositiveRules(rulesFeature, label, table);
                tags_pr = probabilityTags(tagsFeature, label, table);
                tv_pr = probabilityTvTerms(tvTermsFeature, label, table);
                time_pr = probabilityTime(timeFeature, label, table);
                tf_pr = probabilityTf(tfFeature, label, table);

                int sumLabel = 0;
                for (Boolean bool : table.getLabelsList()) {
                        if (bool.equals(label)) {
                                sumLabel++;
                        }
                }
                int labelListSize = table.getLabelsList().size();
                float labelProbability = (float) sumLabel / labelListSize;
                Double bayesProbability = labelProbability * actors_pr * characters_pr
                                * network_pr * rules_pr * tags_pr * tv_pr * time_pr * tf_pr;
                return bayesProbability;
        }

}
```

APPENDIX 2: OOP DESIGN OF THE CODE DEVELOPED

The following table presents the whole collection of the software classes that were developed in this project:

classification package	
Class	**Aim**
ActorsFeature	Checks a given tweet for actor information
CharactersFeature	Checks a given tweet for character information
ConceptFeature	
NetworkTermsFeature	Checks a given tweet for TV channel Information
PositiveRulesFeature	Checks a given tweet for expressions which are frequently used in TV-related tweets
TagsFeature	Checks a given tweet for tags that had been exported from the subtitles of the TV-series
TFeature	Simulates a feature that defines the TF score of the TV-series title
TimeFeature	Checks if a given tweet is published during the airing time of a certain TV-series. (1st SampleSet)
TimeFeature2	Checks if a given tweet is published during the airing time of a certain TV-series. (2nd t SampleSet)
FeatureCharacteristics	Interface that defines a method that the most of the feature classes must have
TVTermsFeature	Checks a given tweet for terms that are used often when talking for television.
Bayes	Calculates Bayes probability
TrainSet	It is responsible for training a set that consists of tweets
FilterTweets	Filters a collection of tweets
RunFiltering	Runs Filtering Class
RocGraphs	Constructs a Roc curve for a certain filtering procedure
BarGraphs	Constructs a Bar graph tha represents the efficiency of the filtering system
Results	Displays Roc and Bar graphs
ConvertToIMDBid	Converts ids of Certain TV-series to the corresponding imdb ids
EvaluateCapitalization	Evaluates the capitalization feature

TagExtractor package	
Class	**Aim**
CalculateIDFscore	Calculates IDF score
CalculateTFIDFscore	Calculates TF-IDF score
CalculateTFscore	Calculates TF score
CosineSimilarity	Produces Similarity between a TV-series and a description
Descriptions	Stores the description of the three examined TV-series as appeared on IMDB.
ExtractTags	Tag Extraction System
Preprocess	Applies Preprocessing in the datasets
RunForAllSeries	Runs the Tag extraction mechanism for all the examined TV-series
Tokenization	Applies tokenization
VisualTags	GUI that presents the results of the tag extraction mechanism

StoreDatasets package	
Class	**Aim**
CsvTool	Reads .csv datasets
StoreDatasets	Stores datasets In database
TestConnection	Tests database connection

TrainingSetCompilation	Compiles a trainingSet from the available tweets dataset

resources package			
Class	**Aim**	**Class**	**Aim**
HibernateUtil	This class is used in order to connect the program with the database	Tweets	Tweet ORM class
TaggedWord	This is a datatype which stores a word and its corresponding part-of-speech type	Transmissions	Transimission ORM class
Tags	Tags ORM class	TrainingSet	Training Set ORM class
TrainedTable	Trained Table ORM class	Subtitles	Subtitles ORM class