Football Transfer Rumours on Twitter: Who and What Should We Believe?

Author: Wiliam Thomas - C1519266

Supervisor: Richard Booth

Module Number: CM3203

Module Title: One Semester Individual Project

Credits: 40

Submission Date: 04/05/2018

Abstract

This research aims to develop a reliable method to compute the reliability of sources posting social media content regarding football transfers in the English Premier League. The first phase of the research looks at the feasibility of implementing Natural Language Processing Techniques in order to process social media posts regarding Football Transfer News in the English Premier League to the formal logic form of 'User U Claims Player P will join Team T". The second phase of the research will concentrate on developing/adapting a suitable algorithm to compute the reliability of each individual source based on their predictive statement regarding football transfers on Twitter during the 2018 January Transfer Window. The result of this project will hopefully demonstrate that it is feasible to have an automated system that can measure the trust of sources on social media which as a result will encourage more accurate news sharing at a time where confidence in news content on social media are very low.

Acknowledgements

I would like to take this opportunity to thank my supervisor Dr Richard Booth for his support and guidance which has been greatly appreciated.

Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

1 Introduction

The last few months has seen a crisis of trust in news outlets, ranging from large multinational news stations to the sharing of social media news articles from freelance journalists.

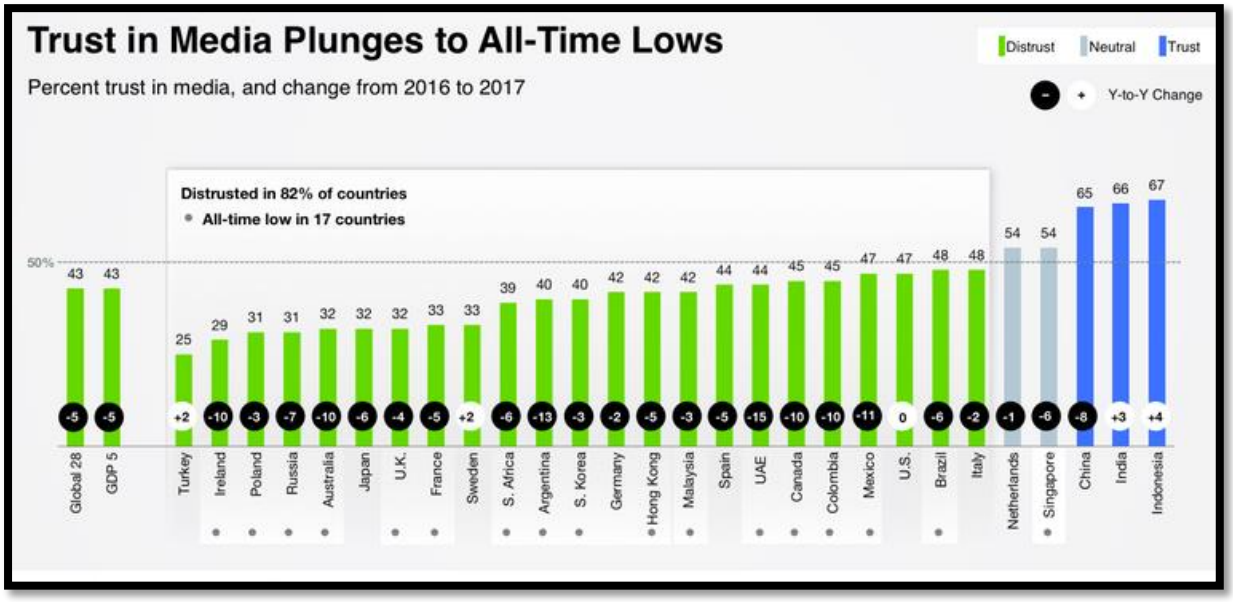

Figure 1 Trust in Media (Edelman, 2017)

There are many theories, claims and controversies regarding the issue, and no one will agree on why the public don't see news as 100% reliable, but this problem isn't a recent problem. The issue of who and who not to trust exists in every profession, industry and social setting. People have accused journalists of lying and misleading the public in the past. But in the last year, it is evident that the general public's view of the media has rapidly deteriorated, with popular phrases such as 'Fake News' being consistently directed at every corner of the industry.

There are 2 main type of news media, traditional news such as newspapers, and television programmes. These means of publishing news has been around for a very long time, and are traditionally controlled by large, often international corporations. While recently, news media can be articles published and shared on the internet, in particular on social media platforms such as Facebook, Twitter and many others. These articles can be shared by anyone and everyone.

While both media have faced increasing distrust from readers, it is evident that people believe less of what they see online on social media platform in favour of traditional news.

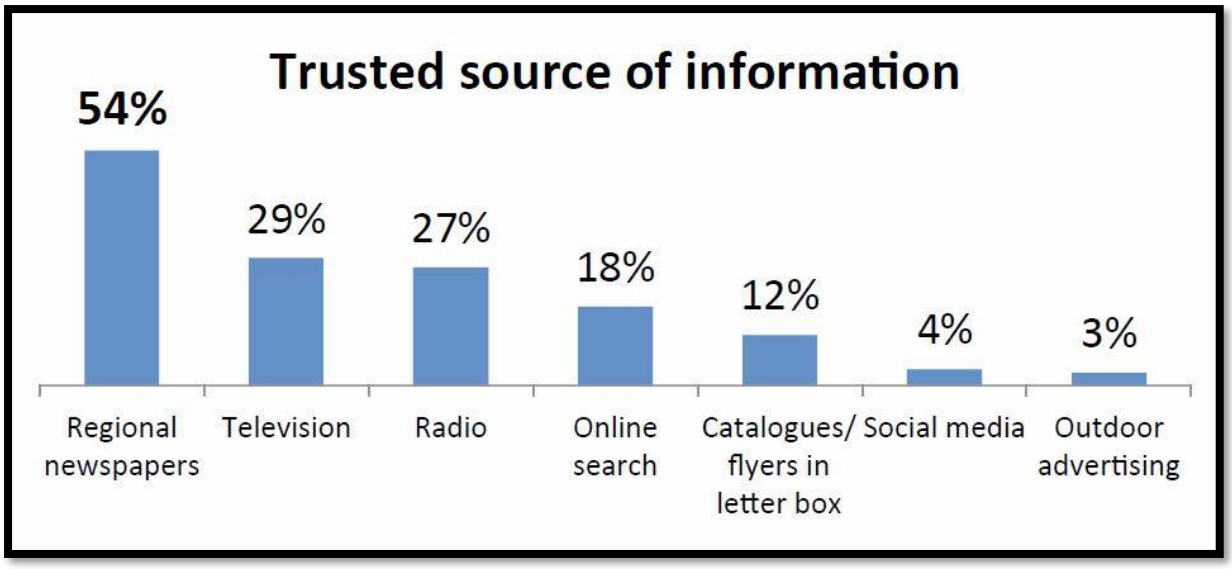

Figure 2 Trusted Sources of Information (Research Now, 2016)

Understandably, a simple reason for this is the fact that the internet is readily available for anyone from any background to publish information. An individual does not need to have a degree from a credible academic institution and does not need to be employed by a reputable news agency to publish information on the world wide web, while traditional news media are more likely to require their journalists to have some credible background before their articles can be published to thousands of readers. It is no doubt that the invention of social media has meant that the internet has enabled millions of news articles and documents to be made available to the public which has in turn allowed the average person to become better aware of what is going on in the world. But, social media has also meant that millions of people around the world are getting fed incorrect news and facts, be that due to inaccurate reporting methods or intentional misinforming in order to enable some third party to gain some advantage in any form as described in the article published by USNI (Carnew, 2017)

Anyone who mildly follows football will appreciate this issue more than most, as it has been around for decades. The English Football League hosts 2 transfer windows every year, this is an opportunity for teams and players alike to conjure up a deal with each other in order to move from one team to another. Of course, the prospect of a team losing its best player will undoubtedly draw the attention of fans of that team, similarly, the rumours of a rival team's best player possibly moving to the team that you follow will also cause great excitement. The excitement around the transfer market is clear to see every year, and this meant that the media have always been quick to jump on the wheel. But, infamously, the news media have not always been accurate, which has led many news organisations to compete against each other for the title of the most reputable source. Ever since the invention of social media, the story has been similar, as an experiment by a journalist with the Galway Advertiser demonstrated. The experiment showed how easy it was for transfer rumours to travel, even if, in this case, the player in question did not exist (Smith, 2017). A simple example would be where 2 different sources tweet about the same player, one source

claims that this player will move to team X while the other source claims the player will move to a different team – team Y. In such a case, who is the reader going to believe?

Clearly, there is a major problem - this is what this project is all about, with the aim of devising a system which can aid the reader when deciding what to believe and what not to believe when reading transfer rumours from thousands of different sources on a social media platform.

This project targets a very niche area of the general issue of who to trust on social media, and it should be seen as a proof of concept to be taken further with the ultimate aim of providing a general-purpose solution to rate the trust and belief of every source and story on social media. This work focuses on using readily available twitter data, along with natural language processing techniques to prepare a suitable dataset to then determine which sources are reliable, and which sources are not.

While overall, the issue in question here is the distrust in social media news, there are many reasons for concentrating on football transfer rumours, which include:

- The abundance of data available and the simplicity of collecting the necessary data
    - Thousands of accounts dedicated to reporting transfer rumours
    - Readily available APIs for collecting and processing posts
- The distinction between which statements are correct and which statements are incorrect is relatively easy to make when compared to other topics
    - At the end of a transfer window, a player will be playing for a single team, this knowledge is given via the official Premier League player register and can then be compared with the prediction of a source.
    - In many other topics, it is very difficult if not impossible to know with complete certainty whether a statement is true or not

At the end of the project, the aim is to have a system which ranks every source by a trust value, and every claim made by a belief value. These values are to be calculated and modified iteratively to simulate how a system would behave in reality i.e. a real-time system which captures tweets from a source making a claim that a player is to move from one team to another, processes that tweet, and modifies the trust and belief values as a result.

2 Background

After identifying this specific problem – the issue of football transfer rumours and specifically who and what should we believe, from a broader general problem of decline in trust in news media, specifically news content on social media platforms. The next stage of the project is to understand the background of the issue, and then research possible technical solutions, by original means and by studying existing solutions to similar problems. Before deciding on the approach to be taken to solve this problem, it is also important to consider the limitations and constraints, which will be useful in the later stages when deciding on the next logical steps to be taken leading on from this project.

2.1 The football transfer market

While the ultimate, long term aim of this research is to have a system to rate the reliability of all news sources, this specific part of the project is all about claims made about the movement of players in the football transfer market. Therefore, it is essential to fully understand what the football transfer market is, and how it works. The article from JLloyd summarises what the football transfer market is very well (JLloyd, 2016).

The football transfer market is just like any other market that deals with the exchange of goods, but in this case the goods are professional football players, and the merchants are football clubs. Throughout a football season, a transfer can only occur during a prearranged window. No transfer can occur outside these periods. There are 2 transfer windows in a season, a summer window and a winter window. The exact dates are to be determined before the season, but these run roughly over the summer months during the offseason of the football league, and throughout the month of January. This project will concentrate on the January window. It is important to understand the true value that the transfer market holds. The graph below shows the economic value of the transfer market

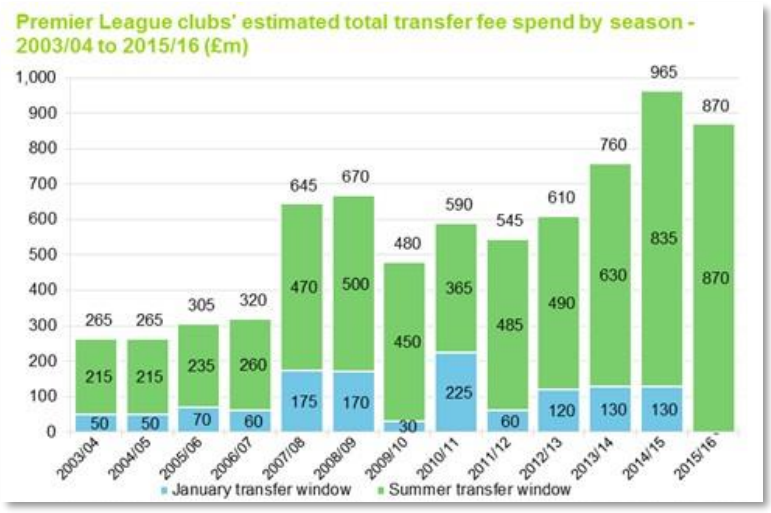

Figure 3 Premier League Transfer Spend (Deloitte, 2015)

Figure 3 shows just how much value and money is in the market, which is important in the case of this project. A market with very little value would suggest it was of little importance, which would eventually mean very little interest and very little data available.

Figure 4 further shows the 'Movement of Goods' within the market, in this case it shows how much players every premier league club have signed in the 2017 summer window. The data was gathered from the official Premier League website (The Premier League, 2017).

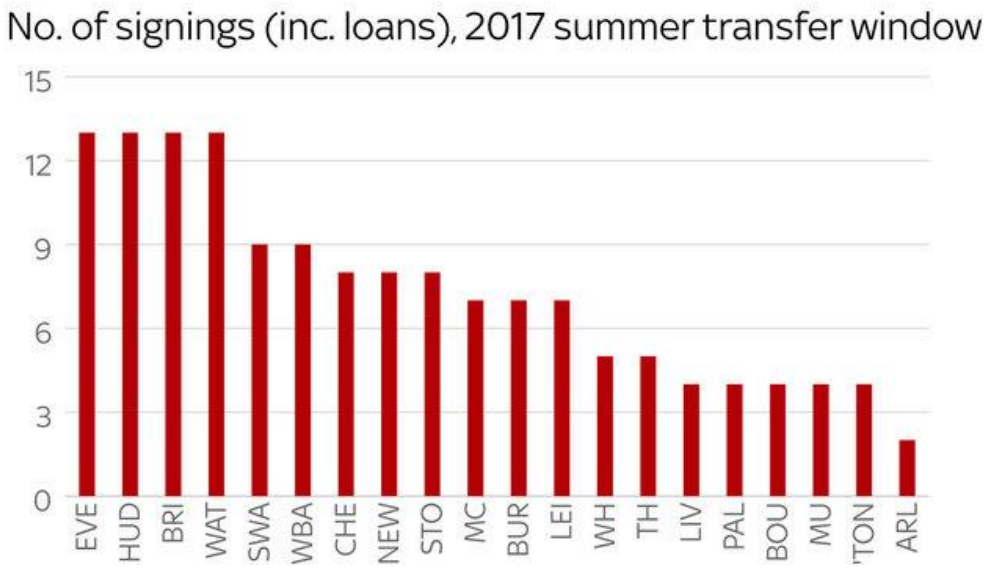

Figure 4 Number of signings by the 20 Premier League Clubs

When attempting to compute the reliability of different sources, it is vital that any project concentrates on a meaningful topic to ensure the availability of data. Football is the most popular sport in the world with an estimated 3.5 billion followers globally according to table 1 below. With such a following and such financial resources being poured into the sport, it would make sense to predict that there will be no shortage of data in this field

rank	Sport	Estimated Fans	**Regional Popularity**
1.	Soccer / Association Football	3.5 Billion	Europe, Africa, Asia, America.
2.	Cricket	2.5 Billion	Asia, Australia, UK.
3.	Field Hockey	2 Billion	Europe, Africa, Asia, Australia.
4.	Tennis	1 Billion	Europe, Asia, America.
5.	Volleyball	900 Million	Europe, Australia, Asia, America.
6.	Table Tennis	850 Million	Europe, Africa, Asia, America.
7.	Baseball	500 Million	America, Japan.
8.	Golf	450 Million	Europe, Asia, America, Canada.
=9	Basketball	400 Million	America.
=9	American Football	400 Million	Europe, Africa, Asia, America, Australia.

Table 1 Sports following around the world (mostpopularsports, 2016)

2.2 Social media platforms

Having concluded that the football transfer market was a good direction for this initial project, it was equally important to choose a suitable social media platform to extract the data from. There were a few key considerations to be taken before deciding which was the best platform to work on. These were

- The popularity of the platform – To obtain credible final results, it is important that plenty of data from many different sources are used.
- Availability of relevant data – A popular platform is of no use to this project if there is little relevant data on the platform. For example, the social media platform LinkedIn is a very popular platform, but is not relevant for this task as there is little to no content regarding football transfers on the platform.
- The available methods to extract suitable data – Important to review the available API's and their ease of use before deciding what platform to use.

Social media is a relatively new invention, but it has rapidly grown to see over 2 billion users as figure 5 below shows.

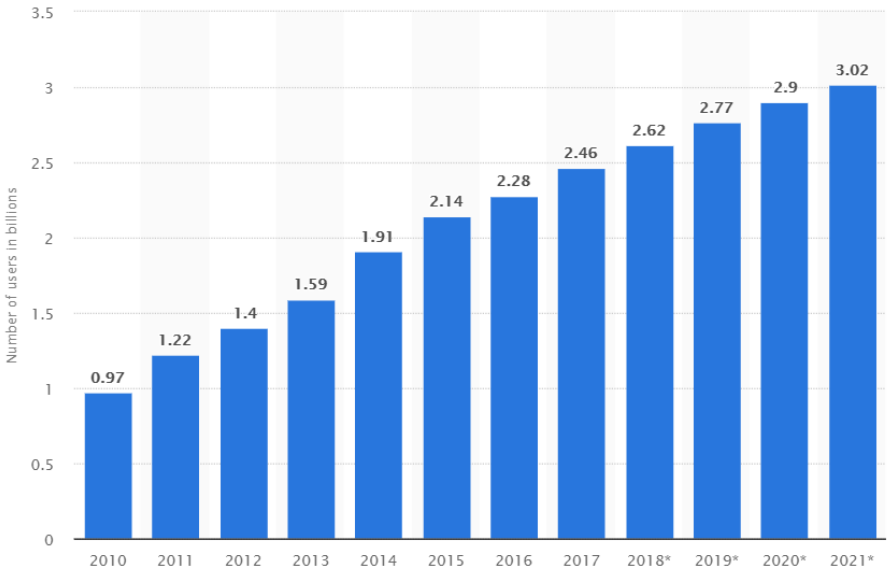

Figure 5 Number of worldwide social network users (Statista, 2017)

The past decade has also seen many new platforms spring up. There are now different platforms specialising in different areas. When deciding what platform to use when preparing relevant data, it is important that the platform is a reputable one. Table 2 below shows the most popular platforms in the UK, along with a short description of the platform itself.

	Description	UK Users	Total Users	Useful information
Facebook:	A social sharing networking site.	32,000,000	1.65 billion	83.6% of Facebook's daily active users live outside the U.S. and Canada.
YouTube:	The top website used for video uploading and viewing.	19,100,000	1,300,000,000	3.25 billion hours of Youtube videos are watched each month.
Twitter:	A Micro-blogging platform.	20,000,000	1.3 billion	50% of users visit the website of a small or medium business they follow.
Instagram:	A photo and video sharing social networking.	14,000,000	500,000,000	80 million photos are shared each day on Instagram. 14 million.
Google+:	A social networking project used to connect with businesses and users.	12,600,000	2,200,000,000	74% of Google+ users are male.
Pinterest:	A popular photo sharing website.	10,300,000		92% of Pinterest users access it through their mobiles.
Snapchat:	Send images and videos with a short life span over an app.	13.6 million	600,000,000	9,000 Snaps are shared each second on the app.
LinkedIn:	B2B platform for networking professionally.	19,000,000	467 million	The most overused word on a LinkedIn profile is "Motivated".
Tumblr:	A popular microblogging platform used to broadcast messages.	9,000,000		69% of Tumblr users are Millennials.
Reddit:	An entertainment, social news and social networking website.	6,600,000	1.3 billion	The average time a Reddit user spends on the site is 16 minutes.

Table 2 List of the most popular Social Media Platforms (Scoial Media Marketing LTD, 2017)

It is important to evaluate the description column of Table 2, as it is no use trying to extract data from a platform such as LinkedIn, which focuses on professional B2B networking. Similarly, while YouTube – a video sharing platform, is very popular, and there may well be sources publishing relevant information on the platform, but extracting the information into a suitable format would be much more difficult than it would on a mainly text based platform.

Data processing on text based sources is undoubtedly easier than on images or videos, and given the many limitations of this project, mainly the time limitation, it would be wise to stick to text based data which rules out some platforms as suitable for the tasks of this project.

Because of the specific topic that is concentrated on in this project, there is a lack of suitable datasets, and with the project requiring substantial amount of data for reliable outcomes, it is vital that a method is able to be implemented which extracts the required data from the chosen platform, therefore the availability and ease of use of different API's for different platforms is a key indicator of whether it is feasible to build up a dataset from the content of a specific platform. Most of the popular platforms offers APIs, and there are many benefits of choosing different APIs over each other as is summarised well in the report 'Top 10 Social APIs' (Katz & Doron, 2015)

Further research into the APIs of different platform revealed the lack of official documents for the APIs of Tumblr and Reddit compared to the abundant amount of documentation for the APIs of Facebook and Twitter.

In reality, the decision was between extracting the data from Facebook, or Twitter. While Facebook in general had many more users, Twitter is the most popular platform for news content, probably because of the structure that Twitter is based on i.e. a micro-blogging

platform when compared to Facebook which focuses more on social connections between friends. Knowing this, and the fact that I have previously used one of Twitter's API in the past, it was a logical choice to concentrate on Twitter as the social media platform to extract the required dataset for this project.

2.3 Technical solution to the identified problem

With trust being such a major issue within social media content, naturally there is a great deal of attention on the issue. But most of the attention and solutions offered has mainly been educational guides containing common principles such as 'Is the account registered for a long period of time', there have been very little attempts to come up with a computational method to solve this issue. There are numerous possible reasons for this, one being the complexity of such a solution for example when a statement is made by a source, who and what is to decide what the correct statement is. In the case of this project, there is a simple solution to this issue, seeing that all elite football clubs have official verified twitter accounts that regularly tweet news, we can take their statement as 100% credible and to be used in the future to compare with the current statements made by different sources. The other possible reason for the lack of computational solution to the problem is the limitations of current Natural Language Processing techniques, however recent advancement in this area has provided a very optimistic future for researchers attempting to provide similar solutions to this project. Evidence of the recent progress of NLP can be seen in the new market created with the release of so called 'Smart Speakers' such as the Amazon Alexa, Google Home (Rouse, 2017).

This project is aiming to use computational means to aid social media users in deciding when to trust different sources and whether or not to believe claims made by different sources. At this point in time, there is no similar system in place. Logically, the project can be split into 3 different parts, the 3 parts being:

- Twitter data extraction and filtration
- Natural Language Processing of data into suitable format for computation
- Computing trustworthy and belief values from resulting data

The aim of this project is to create a system that at this point in time, does not exist. However, the aim is not to find original new ways of completing the steps required to achieve this aim. The availability of a Twitter API means that there is no requirement to come up with an original method to capture Twitter data. In the same sense, Natural Language Processing is a hot topic now, with plenty of research being devoted to this area. This project does not look to expand our capabilities of processing Natural Language, it merely uses existing work to help achieve the original aim of this project. Again, methods of calculating reliability is an area that has been previously researched. This project seeks to utilise the existing work available and not come up with original formulas to compute reliability.

Initially, there seemed to be very little existing work out there, but when splitting the project into the necessary steps to achieve the final aim, it becomes easier to relate the

necessary requirements for this project with existing research, which makes this project more feasible in the time allotted. This project is simply an adaption of many existing work to come up with an original solution to a well identified problem.

In terms of the implementation, the programming language to be used for this project was Python – specifically Python version 3.5.1. The rationale for proceeding with Python was simply that it is the preferred programming language of the author, along with the fact that it had an extensive library of suitable modules i.e. Tweepy the Twitter API, numpy for all mathematical formula that might need to be implemented and a NLP module called NLTK.

2.4 Existing works on similar problems

The first technical task of this project is generating a suitable dataset. Given that the requirement of this dataset is very precise, finding a suitable existing dataset was very unlikely, therefore time was allocated to studying methods to extract the necessary data as opposed to searching for possible existing datasets. This was the part of the project that the author had the most experience with, the experience was gained mainly by following thorough tutorial and guides published by Anthony Sistilli (Sistilli, 2017). The guides published by Sistilli gives a detailed description of the Twitter authentication process and the necessary method calls required to stream live Tweets. Using Sistilli's guide along with the official Tweepy API documents (Roesslein & Joshua, n.d.) allowed for better understanding to complete the required steps to create a Twitter developers account which meant that the necessary authentication data to start extracting data from the platform could be retrieved. The second technical task of this project is to process a dataset of tweets into a formal normalised form to be able to compute the trust and belief values of all sources and claims. Because I have limited experience in NLP, a good proportion of the time available was allotted to background research on NLP as seen in the work plan. This was due to the fact that this part of the project was by far the riskier part. The failure to implement a method to solve the problem of NLP in this context would mean that it would become impossible to apply an algorithm to calculate the necessary values to achieve the results desired. To mitigate the risks of a failure to process the tweets, I came up with several strategies instead of completely relying on one method.

The requirement of this part of the project was to end up with a set of statements which contained 3 variables :

- The User *U:* This variable refers to the Twitter user – the source who is making the statement or the claim regarding a transfer
- The Player *P:* This variable refers to the Player that the user *U* is making a claim about
- The Team *T*: This variable refers to the Team that the User *U* Claims Player *P* will move to by the end of the transfer window

The dataset of tweets is required to be processed in order to come up with a set of statements that will be normalised into a form to convey the following

User ***U*** Claims Player ***P*** will join Team ***T***

Equation 1: Output of Natural Language Processing on an entry in the dataset

The ideal scenario is to essentially combine NLP techniques with ML (Machine Learning). NLP on social media text is a notoriously difficult task as discovered in the paper Noisy Social Media Text (Timothy Baldwin, 2013). Text on social media does not compare well to text in official corpus's, making NLP harder and more complex. Current NLP techniques relies on comparing new text with previously known and processed text. At the moment there are hundreds of well documented texts corpus's, however social media text can be very unpredictable and notoriously filled with misspelt words by conventional standards. It is essentially a language by its own means, with its own specific syntax, a clear example of this is the hashtag symbol, its meaning in formal language is very different from its meaning in social media language, unfortunately, there are many other similar cases making NLP much harder when dealing with Social Media Language.

The aim is to use NLP techniques such as tokenisation and word tagging in order to normalise the Twitter text i.e. remove 'stopwords', symbols and any links to external content. Tag every remaining word in the tweet in order to search the tweet for important content such as the player and the team. Combine this with a ML Technique to classify the text. This is a risky aim due to the time constraint set on the project, the known difficulty of processing social media text, and limited experience in this area. Given that it is an essential part of the project, 2 different methods were proposed should this method fail to provide acceptable result to advance with the project. These were

- A very simple method to simply extract a Player *P* and a Team *T* , and avoid the classification process, which would mean that tweets with different meaning would be taken to mean that the player mentioned will transfer to the team mentioned, even if this was not mentioned in the tweet. This would provide a suitable dataset to carry on with the project at the expense of inaccurate data. The presence of inaccurate data would translate into noisy result, but it would still be possible to demonstrate that an algorithmic computational approach to determining trust and reliability was possible, and given better understanding and more time that a Machine Learning approach to classifying Twitter data would give the reliable data necessary to provide good final results i.e. a list of reliable and trustworthy Twitter sources.
- The second alternative was to manually create a dataset that would mirror real world data. This would allow the project to go ahead past this stage, with accurate data. The only disadvantage of this method would be the limited size of the dataset. Because it would be manually created, it is not feasible to generate a dataset of the scale an automatic computational method would do.

Of course, both of these methods are back up methods and they are not expected to be used. In the event that they did have to be used, a big part of this project would be a failure,

however it would still be possible complete the remaining parts of the project i.e. determine if an algorithmic approach to computing reliability is possible.

NLP is a very hot topic at the moment, therefore there is no shortage of resources available on the topic, but in order to truly understand it's possibilities and limitations at present, it was important to understand and research the history of NLP. The paper on the Advances in NLP summarises the topic very well, with linguistic and technical theory behind NLP explained very well (Hirschberg & Manning, 2016) The bulk of the implementation work will be done using the NLTK module for Python. A fair amount of work has been done on this module along with a guidance book published by the creators (Bird, et al., 2009) This allows for the utilisation of readymade and tested NLP techniques to be used on the dataset, without such existing methods, this project would not be feasible given the time constraint. In addition to NLP techniques a form of ML (Machine Learning) must be employed in order to classify different tweets according to their intended meaning. A tweet might mention a player and a separate team, but it might not indicate a transfer to that team. It is important for the system to be able to understand the intended meaning of a tweet and classify the tweet accordingly. There are many different possibilities to achieve this, and previous research into the different possibilities already exist allowing me to make an informed decision of what method is the most suitable in this case. (Maglogiannis & Karpouzis, 2007)

The final part of the project is to adapt and apply some existing algorithm to compute trust and belief values for sources and claims. There are several existing works on similar algorithms. Examples are are from J Pasternack and D Roth who between them has formulated a number of equations based on many different algorithms that pre-existed to deal with the issue of computing trust and belief (Pasternack & Roth, 2010) . Also possible, are conventional statistical methods such as Bayesian Networks that can be adapted to suit the needs of this project (Langseth & Portinale, 2007)

2.5 Proposed Solution

This section provides the necessary background information for the reader on the 3 sections of the project, before moving on to describe the approach and implementation

2.5.1 Generating suitable dataset

This section is all about using the Tweepy module, which is an extension of the Twitter streaming API to make it easier to stream twitter data on Python by handling the necessary connections i.e. authentication, session initiation, reading incoming messages etc. The first step in using the API is to create a developers account on Twitter which will generate the necessary authentication credentials to make the connection, these are:

- Consumer Key
- Consumer Secret
- Access Token
- Access Token Secret

The next step is to install the Tweepy module. It is compatible with a range of Python versions, more information can be found in the official Tweepy documentation (Roesslein & Joshua, n.d.)

It is not necessary to know exactly the code behind how the Streaming API works and its communication with the Tweepy module, however readers can refer to the API official documentation for more information (Twitter, 2018)

It is also essential for the user to briefly appreciate what the JSON format is, as the output from the streaming process are JSON format files. For detailed information about the format, readers can refer to the official ECMA publication on JSON (ECMA International, 2017). In short, JSON is a relatively new format which aims to make it a lightweight interchangeable format and easy for humans and machines to understand, read and write.

2.5.2 Natural Language Processing and Machine Learning

NLP is an area of Computer Science concerned with the interactions between human languages and machines. It is specifically involved with speech recognition and natural language understanding. This project is only concerned with natural language understanding. Before understanding the technical techniques of NLP, it is important to understand the linguistic aspects. This means being familiar with terms such as Tokens, Lexical Resources, Stop words, Noun, Verbs etc. If the reader is not familiar with these terms, it is advisable to read the first 5 chapters of the NLP with Python book (Bird, et al., 2009) Furthermore, definition of some terms to be used throughout this report is included in the Glossary for reference.

In terms of this project, the main goal of the NLP is to strip the data of certain types of words, such as named entities, URL's, symbols etc. This will make it easier for a ML algorithm to classify whether the text is indeed referring to a transfer rumour or not. The main techniques used to achieve this are:

- Lexical Analysis (or commonly known as Tokenization)
- Terminology Extraction
- Part-of-Speech (POS) Tagging
- Lemmatization
- Named Entity Recognition

These are all common NLP techniques, and pre-defined methods are available in the NLTK module for Python. Combining all these techniques will be sufficient to process the text to the necessary format.

Lexical Analysis or Tokenization will essentially break a stream of text into useful tokens, in the context of this project, tokens essentially means natural language words amongst other Social Media language such as URL's and symbols like the hashtag for example. This is the pre-processing stage of NLP and is an essential task.

Terminology Extraction is a broad term describing the filtration of text. This is an important part of the NLP process in this project as social media text regularly contain URL's to external content, symbols with no semantic value to natural language such as the hashtag or the @ symbol which denotes retweets and replies etc. This process also removes any stop words in the text. Stop words are not universally defined, and there can be a variation between what is defined as a stop word. For this project, for the sake of simplicity, I will use the corpus of stop words already present in the NLTK datasets. Terminology Extraction uses Regular Expressions as a means of filtering out URL's and symbols from the text (Goyvaerts & Jan, 2016)

Next stage in the process is to tag the text. Tagging the text refers to the notion of marking up a word in the text as corresponding to a particular part of speech that is based on both its context and definition. These tags are displayed as abbreviations when using the NLTK module. For explanation regarding what these abbreviations mean, refer to appendix A (Bird, et al., 2009) The reason for focusing on the context of the word in addition to its definition is because a word can be tagged differently according to its context, as shown in the simple example using the word 'Race' in table 3 below.

Part of Speech	**Text**	**Explanation**
Noun	The Jockey won the *race*	A commentator describes a jockey winning a competition
Verb	I *race* horses	The jockey describes what is his profession

Table 3 : Ambiguity in Part of Speech

Lemmatization is the process of determining the Lemma of a word. In simple terms it means to determine the base word, a word can have many different variations i.e. raced, racing, races are variations of the word race. The lemma of races is race. Lemmatization is the computational process of determining this. This technique is similar to a different NLP technique call Stemming. Their end goals are the same, however Lemmatization is a better method of reaching that end goals, as it considers the context of the word and its meaning whereas Stemming does not, which can lead to noisy results.

The final technique to be used in this project is a technique called Named Entity Recognition. This is all about recognizing tokens in the text. It compares tokens to a text corpus with the aim of finding a named entity, this is used in conjunction with Part of Speech Tagging to increase efficiency i.e. only compare nouns against the corpus of named entities.

Figure 6 below shows a diagrammatic representation of the whole process

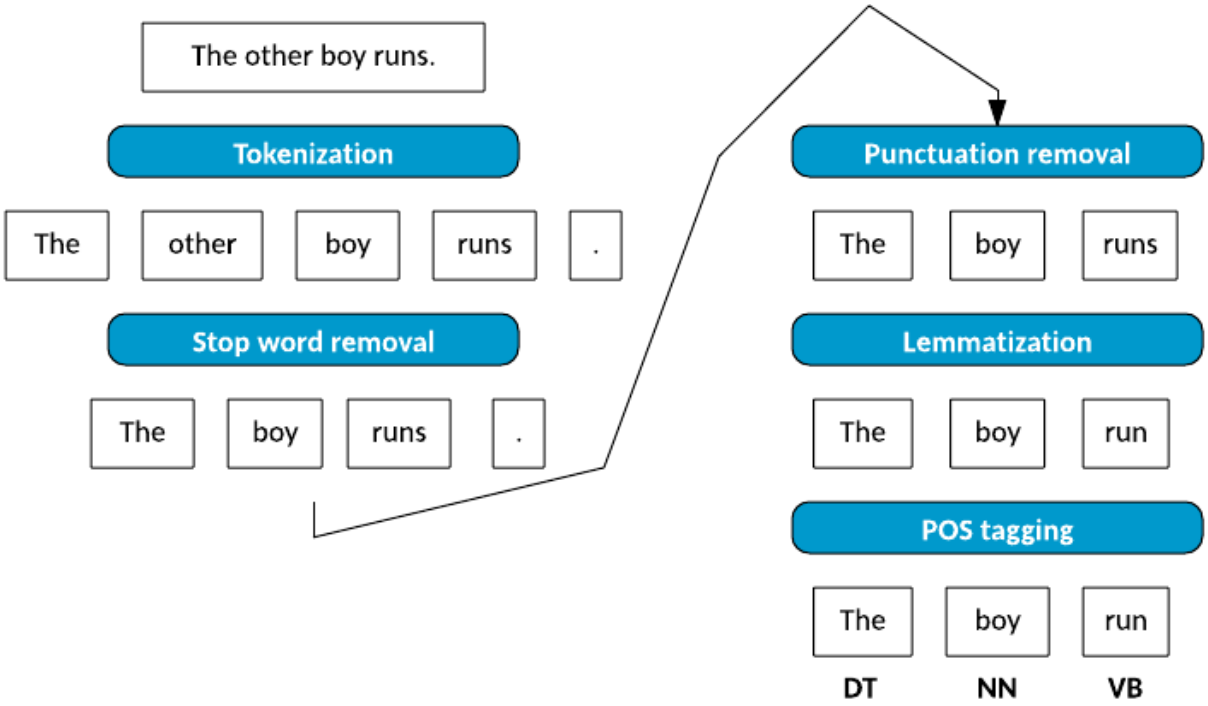

Figure 6 : Diagrammatic visualisation of the NLP process (Jones, 2017)

The next stage would be ML. ML, like NLP is a hot topic with many resources being allotted to the research of new and improved ML methods. There are many resources available on this topic. The task required of this project to complete the requirements set out is a well described problem, and is studied in detail by Fabrizo Sebastiani in his paper on Automated Text Categorization (Sebastiani, 2002) The paper defines what is required of a method in order to categorise text according to a binary value, which is sufficient for this project i.e. a tweet referring to a transfer can be considered to be True or False. The paper recognised the steps required for a ML approach to tackle the categorisation problem. Firstly, it recognised the need for 3 datasets, which were:

- Training Data
- Test Data
- Application Data

The training data is a dataset that is usually prepared by a human or prepared in a semi-automatic way. It is a set of texts that has been pre- categorised. Test data is basically a segment of the Training data that is used to confirm the accuracy of the ML model. Finally, Application Data is simply the uncategorized data that needs to be categorized.

Classification is a specific task of ML. It relies on training a classifier i.e. feeding a classifier the training data to allow it to understand the difference between categories which in turn allows it to decide what is the meaning of different texts.

The classifier itself can interpret the text in many different ways according to the algorithm it is based on. The NLTK Python module offers many different algorithms to base the

classifier on, below is a short summary on each of the algorithms on offer in the NLTK Module along with the formula used and some of its advantages and disadvantages.

Decision Tree (J.R.Quinlan, 1986)

A Decision Tree is a Tree structure containing the standard elements of a tree object. Each internal node of the Tree refers to a test on an attribute, each branch of the Tree refers to a possible outcome of the test, and every leaf node of the Tree is a class label.

The main advantages of a Decision Tree is that it does not require any domain knowledge and that it is very easy to comprehend, however crucially, one of its biggest disadvantage is that it is not a method that scales well. Preparing a decision tree that has many branches is complex and time consuming. It has a high variance and it is a method that is quite dependent on certain initial variables such as the root node. It requires careful manipulation to see good results.

Maximum Entropy Modelling (Berger, et al., 1998)

A classifier model which is based on the maximum entropy modelling framework (E.T.Jaynes, 1957). The algorithm is based on calculating the entropy of different variables as defined in equation 2 below

$$H(X) = -\sum_{k \geq 1} p_k \log p_k$$

Equation 2 : Equation for the Entropy of X (P_k is the Probability of the event k)

This method goes hand in hand with the Decision Tree. Once a Decision Tree is built, it is relatively straight forward to get good reliable classification results. However the main issue is the building of the tree and what do you use as the root node to start the process. Entropy modelling is a method to solve this problem, which it does. It is a remarkable improvement over randomly choosing the root node, however in order to ensure you have a tree which is able to classify categories as efficient as possible, you will need to perform this algorithm on all possible decisions i.e. it essentially becomes a search problem, which is not the best use of computational resource.

Naïve Bayes

Bayes theorem is perhaps one of the best known mathematical theories. Although it's history and origins is contested, it is named after the Statistician Thomas Bayes (Stigler, 1983). The Naïve Bayes classifier takes on a probabilistic approach to categorising data. It begins by calculating the prior probability for each category, this value is calculated using the training set. This is then combined with the contribution of a particular feature – In this case a specific word that is not a named entity in order to compute a likelihood estimate for each category. This means that for all text the algorithm runs against, it starts off being biased that it belongs to a particular category – which is pre-calculated on the training set.

From here, individual words from a new text contribute to the overall decision by moving the decision to the direction of the category that mostly contains the individual word of that text. Individual words (Features) can work against each other i.e. one word which usually occurs in one category of texts can be conflicted by a different word usually occurring in a different category. The algorithm then computes the probability of the text belonging to the different category.

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Equation 3 : Bayes Theorem

Naïve Bayes is a heavily simplified algorithm. It is not a very realistic approach. It has some assumptions that does not generalise well in a realistic environment. It chooses the most likely category for a tweet(text) under the assumption that every tweet is created by first taking into account the category, and that the wording in that tweet are entirely independent of each other. In simple terms, it assumes that words in a sentence are not connected what so ever, as if a sentence is generated randomly – but with meaning. Of course, removing some 'unnecessary' words will greatly reduce the impact this has on our classifier. Unnecessary words can be viewed as words that are likely to occur in all texts regardless of the topic, such as stopwords etc. In regard to the Bayes Theorem used in the algorithm (See Equation 3 above)

- C : Category
- A : Word

By assigning the different available categories to the equation and computing the result, we can then choose the category that maximises the result. The main benefit of this method is the speed and efficiency, it is also an intuitive method that is very easily understood, however, it is not the most accurate algorithm available due to its unrealistic assumptions.

ML is not a flawless technique, and it is known to have many problems. Before deciding on training data and algorithm, it is important to know what such problems are. Example of some common problems are

- Overfitting: A model is overfitting if it models the training data too well for example - It classifies the test data correctly 90% of the times, but when it comes to new unseen data, it only classifies it correctly 50%. Therefore, the model doesn't generalise well from training data to unseen data. The reasons for this is that the model captures the noise of the data. This tends to happen when the algorithm shows low bias but high variance.
- Underfitting: A model is underfitting if it cannot capture the underlying trend of the data i.e. when the model does not fit the data well enough because it is too simple to model the complexity of the data. This tends to happen when the algorithm shows low variance but high bias.

When deciding on the best algorithm for the purposes of the project, it is equally important to consider the nature of the training set before deciding on the final algorithm, exactly what to consider is mentioned in detail in chapter 3.

2.5.3 Computing Reliability Values

The final part of the project is all about what to do with the output. There are many available algorithms that deal with reliability based on many different factors and this part of the project is all about evaluating which algorithm works best for this project. Previous work has been made in this area, notably by J Pasternack and D Roth in their paper on Knowing what to Believe (Pasternack & Roth, 2010). In their work, they proposed many equations that were derived from different existing equations such as Hubs and Authorities (Kleinberg, 1999), TruthFinder (Yin, et al., 2008). All of which are very suitable for this project since they focus on the relationship between sources and a claim, and how to model trust and belief before it can be seen with 100% certainty if a claim was true or not.

The limitation of the methods proposed by Pasternack and Roth for this project was the fact that it doesn't incorporate a method to calculate the trust of a source if a claim they made has turned into a reality (or it is known with certainty that it never will). In our case, a transfer can occur mid window, and it would be beneficial to use this knowledge, which would give a better trust estimate for a source. The absolute simplest method of doing this is to take a percentage value of the correct predictions versus the total prediction made. Of course, this method has so many flaws that it shouldn't even be considered. But a method derived from this method is feasible. For example, a source that has correctly predicted 9/10 events shouldn't be considered as reliable as a source that has correctly predicted 900/1000 events. A method that is also based on the total amount of predictions made would yield a much better method.

In order to come up with a suitable algorithm, it is necessary to adapt existing methods, and connect them with new original methods that better suits the needs of this project. This part of the project will rely on some trial and error of different algorithms.

2.6 Limitations and constraints

Unfortunately, this project is severely limited by a strict time constraint, this translates to many other constraints and assumptions within the project. Some constraints were only realised during the implementation, while some constraints were made clear in the approach stage. Some of the main constraints of the project are:

- The project focuses on the English Premier League only, any movement between players not currently playing for a EPL team will not be picked up
- The project will only process tweets in English, it will filter out all tweets in any other language
- The data processing will be limited to a single statement within a single tweet, if many players and/or many teams are mentioned in a tweet, the tweet will be ignored

- Similarly, any tweet mentioning a statement that is inferred from a separate statement will be ignored. For example, Player P_1 will move to Team T_1 if Player P_2 moves to Team T_2
- The project will work only on data extracted during the 2018 January Transfer Window, any statement made by twitter sources outside this window will not have been picked up, and therefore will not be taken into account when computing their reliability

As has been repeated in the introduction, this project does not aim to provide a fully functioning social media reliability tracker ready to be deployed to the masses, this is merely the first step towards the direction of achieving that aim. The constraints listed above are there to simplify the project, especially the NLP and ML tasks. An entire thesis can be written on the different available data that can be extracted, computed and inferred from a single tweet, this project focuses on only the basics. Of course, any future work would have to do more to extract more data and a more complete statement that better represents what the source is trying to say. There will be a whole chapter dedicated to the future work to advance this project. As constraints appear in the project, these will be fully discussed and justified in the approach and the implementation chapters later in the report.

2.7 Ethical Considerations

This project makes use of publicly available data on the social media platform Twitter on a large scale. It will stream all the relevant tweets and capture the tweets deemed suitable without any explicit permission from each user. However, all Twitter user will have been made aware of the implication of performing any action on twitter such as publishing a tweet, or retweeting a different tweet etc. These implications are all explained in detail in Twitter's Privacy Policy (Twitter Inc., 2017). In a nutshell, the terms and conditions mean that all Twitter users are consenting for their tweets to be used by any third-party organisation.

However, in order to access this data, attention must be directed towards the developer agreement (Twitter Inc., 2017).

To comply with the school's individual ethical policy, a number of steps have been taken. These are

- Data will be securely stored using a medium approved by the School
- Results will be anonymised to protect the identity of all sources whose data have been captured in this project

3 Approach

The introduction chapter explained the overall picture of the problem, targeted a specific area of interest which shares the general problem and gave a brief outline of what the technical solution to the problem is. The Background chapter explained some areas of the project the reader might be unaware about, along with some basic introduction to some of the technical concepts that the implementation of this project will use. This chapter will introduce a plan of implementation. The approach was logically split into 3 different sections. These 3 areas have different requirements and tasks which are fully explained in this chapter.

3.1 Data Capture

The first task presented to us is to solve the problem of capturing the necessary data to perform the processing steps and compute whether the source sharing this data is reliable. Given the lack of any available dataset containing tweets about football transfers, a script had to be implemented to stream twitter data, extract the data that was suitable and convert it into a suitable format. This was made feasible by the Python Module Tweepy

3.1.1 Requirements of Data Capture

The requirements of this section are dictated by how the data is required to be formatted at the end of the next section. Next section requires a dataset of captured tweets that claims a transfer of any player currently playing for an EPL Club will occur. Each entry of the dataset is to include the following:

- Date & Time of the publication of the tweet – This is a requirement for the task of computing the belief of a claim. A claim made by a source after a transfer has been made should not be classed as a claim, as it is no longer a claim, the source is merely stating a fact
- Source username & id – These are identifiers to keep track of what source has made what claim
- Text – This is the actual text that was published in the tweet by a source

It is important to recap that a transfer window only occurs twice annually, one during the summer months, and one during the winter. This project aims to capture all the relevant tweets about the winter (January) transfer window.

Using Twitter's streaming API along with the filter method will mean that all tweets containing one or more of the keywords in the list passed as a parameter to the stream.filter method will be captured by the Stream Object. The Object will then print all the metadata and textual data contained within the specific tweet into a JSON entry. A typical JSON entry will contain a wealth of data that is not required for the purposes of this project, therefore a further processing step is necessary to filter out all the data that is not required. The final goal is to have a CSV file, where rows constitute a tweet from a source and each row will contain the Date & Time of the tweet, the username & id of the source,

and the text that was tweeted. The example below in table 4 demonstrates how the output should look like

Date & Time	User ID	Username	Tweet
07/01/2018 16:43:01	433259	50FootballNews	Sergio Aguero set to move to Man United in shock move #Aguero #ManUtd
07/01/2018 16:43:03	226597	WBNews	Romelu Lukaku set to miss United's next game against Everton

Table 4 : Example of the desired output

3.1.2 Design of Data Capture

The requirement of the Data Capture is to be fulfilled via a simple Python Script using Tweepy. It is a very process due to the fact that the Tweepy module handles all the complex methods such as authentication, session management, routing and so on. The script is mostly a case of passing on the parameters.

The API works by essentially creating an indefinitely long HTTP request, and pushing the response back to the client. Figure 7 below shows a graphical representation of the process

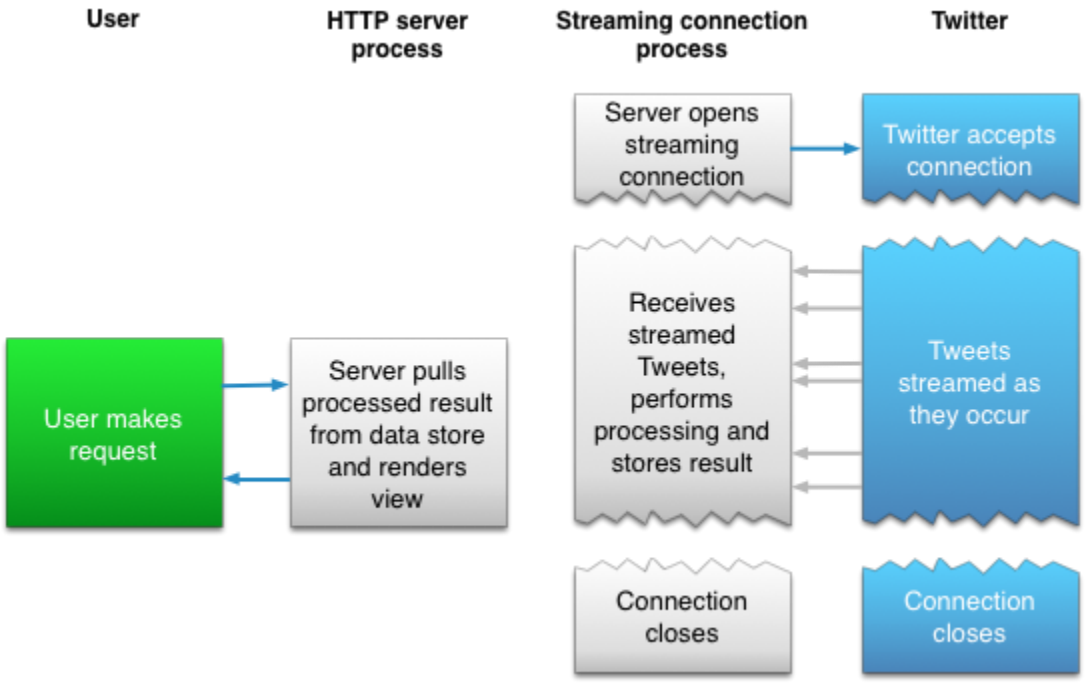

Figure 7 : Streaming Process (dev.twitter.com/docs/streaming-apis)

In terms of implementation, it is a straight forward task with very few lines of code. It will simply be a case of creating a OAuthHandler object with the relevant authentication credential passed as parameters which handles all the authentication aspects. Along with a stream object which will also include a method that will filter out retweets. The stream object will use the filter method, which means that the object will only return tweets

containing any keywords within the list passed on as a parameter. The list will contain the names of all 772 players officially registered with a club in the EPL.

The output of such a script would be a list of JSON objects, where a single object constitutes a single tweet. The next step is to convert this output into a suitable output for the purposes of this project. This means a simple code to manipulate each JSON entry and extract the necessary information into a list, which in turn will be written as a row in a CSV file, where each column in the row contains the information required as set out in the requirements. This file will then be saved, and the final result will be a CSV file containing rows of tweet's, along with the date and user identification. This will be the main dataset to conduct the project on.

3.2 NLP & ML Approach

This section of the project, as previously mentioned is by far the riskiest. Therefore, it made sense to devote more time to this section than the other two. Before designing and implementing a solution to this problem, a large amount of background reading and work was required, which was described in the previous chapter.

3.2.1 Requirements of the NLP & ML approach

In the simplest of terms, the requirements of this section of the project is to transform the dataset that was the output of the previous section, into a list of formal logic statements that reads:

User ***U*** Claims Player ***P*** will join Team ***T***

In order to arrive at this output, there are numerous steps to take. It will take a combination of NLP techniques and applying a ML classification algorithm.

The first step of this section is to identify the 3 variables – The user **U**, the player **P** and the team **T**. Identifying the user is a simple task, as it is captioned in the row where the text is retrieved from. However, identifying the player and team will require some NLP and string manipulation work. If no specific player or team have been identified by the end of this stage, then there is no need to proceed further with the ML classification as the tweet will be discarded, which will save time and resources.

To apply a ML Classification algorithm, the text will need to be divided into so called 'Features'. In the context of this project, features refer to specific words used in the text. To avoid processing words which are not important and does not add any specific content to the text, a NLP approach is required to strip the text of these words, and output the words which add context and meaning to the text. Examples of the words that need to be stripped from the text are:

- Stopwords
- Symbols
- URL's

An example of the desired output is shown below in table 5

Pre-Processed text	Output
Manchester City eye West Brom's Johnny Evans in a transfer that could be worth over 25million http://mancitynews.com/11235	**Manchester City** eye **West Brom Johnny Evans** transfer could worth over 25million

Table 5 : Example of required output after NLP of a text

Before a text can be classified using a ML algorithm, a specific algorithm needs to be decided on. Furthermore, once the specific algorithm is decided on, the classifier must be trained to distinguish between suitable and unsuitable tweets. The theory behind this is explained in chapter 2. When deciding between the most suitable algorithm for this project there are two main points to consider:

- How effective the algorithm is (How many tweets it categorises correctly)
- How fast is the algorithm (An algorithm that is always correct is no good if it takes weeks to run)

As is true with almost every decision regarding the best algorithm, it is a fine line between balancing the effectiveness of the algorithm against the speed at that it performs. One aspect will always be sacrificed for another.

Perhaps more important, is the specific points to consider that are unique to this project, such as:

- No pre-existing suitable training dataset. This will mean that a new dataset will have to be generated which could prove to be difficult / tedious as they will have to be manually categorised. The point here is that, given the time constraint on the project, the training set is likely to be small.
- The ultimate aim of this type of project in the future is to have a real-time system running as the data comes in. However, for this project, all the processing is done after all the data has come in. The main dataset is likely to be very large, therefore the run time of the algorithm is to be of great importance to avoid waiting for hours or possibly days for the system to run through all the data gathered.

Generating a dataset to train the classifier can be problematic. With the absence of any existing categorised training set suitable to the context of the project, there was only 2 real options available, which were to

- Take random samples from the main dataset and manually label them.
- Capture tweets from a Twitter account where all tweets are known to include only transfer news or only non-transfer news.

There are benefits and drawbacks to both methods. Taking random samples from the main dataset is the method which will best resemble the main dataset if done correctly, however there is always a risk of choosing a poor sample. A diverse sample is very important, as there are thousands of sources, each with different informal writing styles, a poor sample might

concentrate on sources with similar writing styles which will mean the classifier will perform poorly on some sources and better on other sources, causing the final reliability scores to be biased. This method is also very tedious – it requires manually labelling the samples, which is time consuming. A different method would be to extract tweets from a Twitter account that posts nothing but transfer news and a separate Twitter account which posts football news but not transfer news. This method would be much quicker, and would mean a larger training data. However, it might not truly represent the entire dataset, as it only reflect what constitutes a transfer tweet from the perspective of a single user and might cause the classifier to perform poorly for some sources who's writing style differ significantly. The best way forward in this case depends on the chosen algorithm, as some algorithms perform better than each other given a much smaller dataset.

There are a couple of different algorithms available on the NLTK platform. They are explained in detail in the second chapter. However, it was not discussed which was the most suitable for this project. Given the specific circumstances of this particular project and the background knowledge on different classification algorithms, it is clear that the most suitable algorithm for this project is the Naïve Bayes algorithm. The reasons for this are:

- Simplicity of the algorithm – Given the time constraints of the project, it is not feasible to spend too much time on building a classification model.
- Limited training data available – There is limited means of gathering a sufficiently sized training data. Therefore, it is crucial that the algorithm can keep up its performance level, or at least ensure its performance does not drastically drop given a relatively small training set.
- Performance speed – The classifier will be responsible for classifying hundreds of thousands of tweets in one go. The feasibility of the whole classification process depends on the speed of the process itself.

Naïve Bayes' main advantages are:

- Simplicity. It is among the simplest classification algorithm available relying on a simple formula as explained in the background section. Because of this simplicity, it is very easy to set up and train a Naïve Bayes classifier and does not require tedious configuration of the classifier to optimise result
- Naïve Bayes classifier is a high bias / low variance classifier, which makes it less vulnerable to overfitting problems when dealing with a small training set. This gives it a big advantage when compared with other algorithms, however this advantage is quickly ruled out once the training set grows. However, for this project, training data is likely to be small
- Naïve Bayes is a simple probabilistic classifier with an assumption that all features are independent. It is not a complex process and therefore not very heavy, computation wise. Compared with most classifiers, it is not an intensive classifier.

As is the same for any algorithm, there are benefits and drawbacks and it's all about compromising and finding the best fit for a particular problem. Naïve Bayes has, as mentioned, a clear constraint, in that it assumes conditional independence of the features. In the context of text classification, this essentially means:

- Given the category of a text, the probability of any word present within this text is independent of the probability that a different word is present within the text
- The probability of a word occurring within a text is independent of its location within the text

The result of these assumptions would render the Naïve Bayes algorithm useless in many ML applications, however, when it comes to text classification, these independent assumptions are not a world apart. The probability of a word being in a text IS roughly independent of another word being in the text. Of course, there are some exceptions, a simple example would be that the probability of the words 'To' and 'From' appearing in a text would slightly increase given the knowledge that the word 'Move' appears in the text ('Move to' / 'Move from' etc). But this is by no means a certainty, therefore the assumption is not completely inaccurate, however it is not fully accurate either. Similarly, for the second point, it is not difficult to find a case where this is incorrect, for example a common technique used when publishing new news on social media is to start the publication with a specific word such as 'Breaking', therefore the probability of a word will depend on its location within the text. However, given the word 'Transfer', is it possible to predict where its location within the text is? This is just 1 example of many, and proves that although Naïve assumptions are not realistic in any sense, in the context of word categorisation, it is possible to roughly adhere to the assumptions.

It is easy to theorise which algorithm would perform better and the reasons behind it, but the best way to decide is to base the decision on experimental data. Luckily, there is an abundance of existing work comparing different algorithms and their performance in different circumstances. The work done by Trivedi provides good data which suggests Naïve Bayes works equally as well as a decision tree approach and a vector approach using the SVM algorithm, with average success rates on the testing set roughly identical. (M. Trivedi, 2015). His work seemed to arrive to the general consensus when it came to experimental results. Given what we know about the Naïve Bayes algorithm, and that it evidently is performing as well as other more modern and complex classifiers, for the purposes of this project, it was a logical move to implement a Naïve Bayes classifier.

All that is required of the classifier is to be reasonable effective in distinguishing between transfer news and non-transfer news. A suitably labelled dataset will be required to train the classifier to do this. This poses a problem as there is no suitable dataset available. In order to come up with a dataset, it was important to note the following requirements for a Naïve Bayes training set listed in table 6 below:

Requirement	Explanation	Example
A dataset containing two columns – one for the text itself and one to denote the category of that text	Naïve Bayes must know the category of texts in the training set in order to be trained for future classifications	Romelu Lukaku set to move to Arsenal in shock transfer → transfernews Romelu Lukaku scores 2 goals tonight as before he is substituted for Manchester United → normalnews
A function which takes the text in the dataset and processes it into a tuple which contained the list of all features and whether they appear in the text or not along with the category	The parameters of the classification & train function of a classifier is a tuple which is formatted as noted	Text = ([set:False,sign:False,move:True, transfer:False…],transfernews)

Table 6 : Requirements of the Training Set

The third dataset that is often used when applying ML techniques is the testing set, which is very useful as it gives an estimate of how accurate the classifier is. In order to have a value that will truly reflect how accurate the classifier is, it would be wise to take a random sample of the main data, and manually categorise these sample before testing the accuracy of the model on these samples. This will give a good indication of how accurate the classifier is. Figure 8 below is a graphical summary of the ML process:

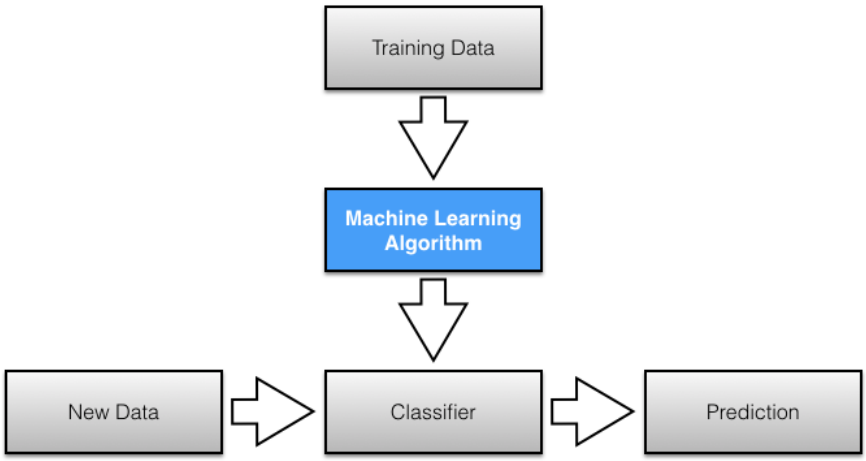

Figure 8 : Graphical Summary of the ML Process

3.2.2 Design of NLP & ML Approach

This stage of the project is considerably more complex than the Data Capture stage. Before going in to detail on the specific designs and algorithms of the different methods, a simple flow chart is seen below (Figure 9) Which summarises the whole process (Note that it is a rough design and does not contain validation methods)

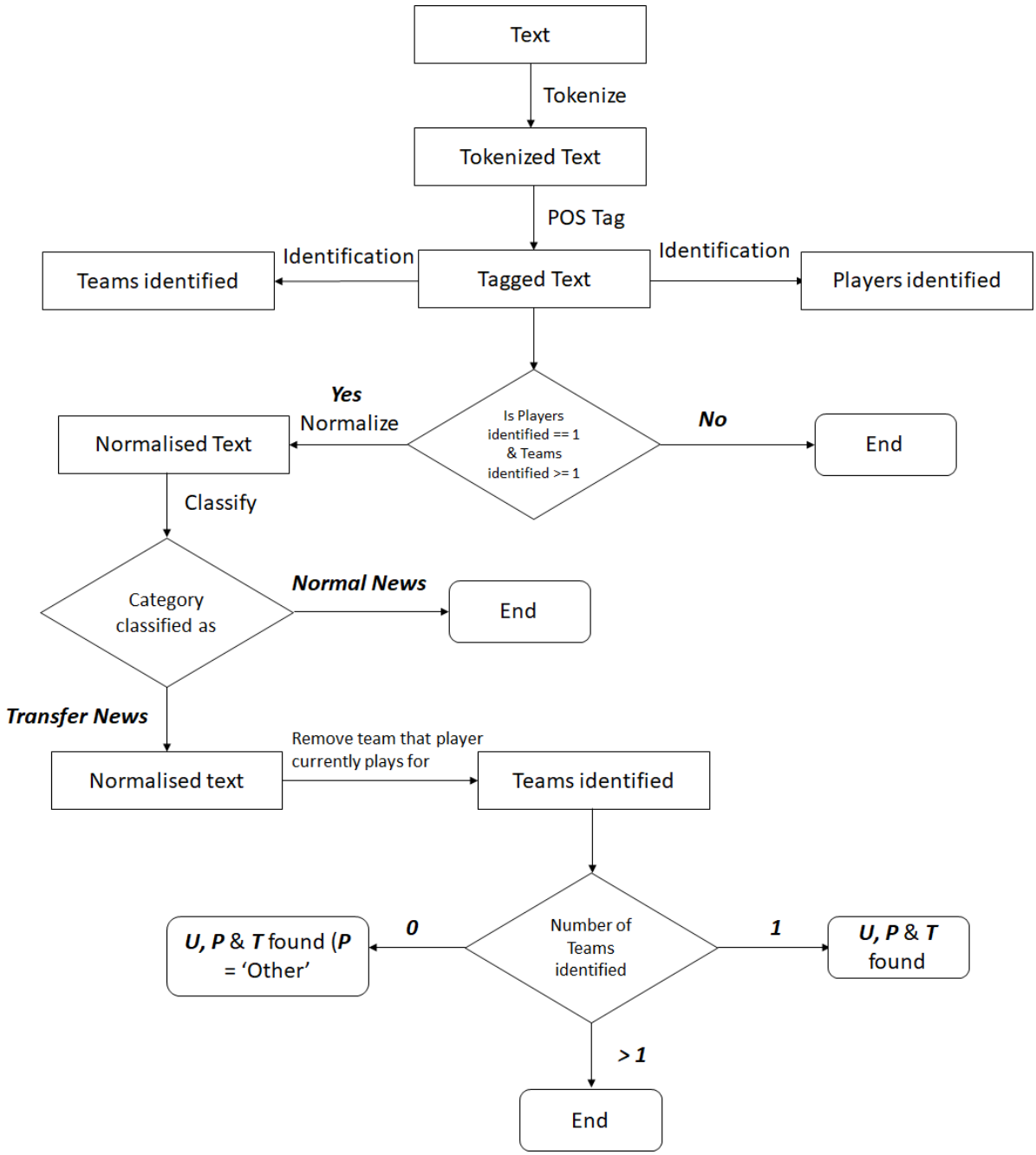

Figure 9 : Flowchart of the NLP & ML Process

The first steps in the process relies on method calls from the NLTK module, these are the:

- Word_tokenize() – Splits a string into a list of smaller strings where each smaller string is an individual word in the original string
- Nltk.pos.tag() – Creates a list of tuples, where each tuple contains the substring passed to the method, and an abbreviation referring to the part of speech tag for the substring i.e. NNP refers to a Proper Noun. For the list which explains each abbreviation, refer to Appendix A

After both these steps, we will have the list of tuples, which will make it easier to process and perform the next steps on it. The next step will be to identify the team and the player mentioned in the tweet. This requires some basic string manipulation. Because the POS_TAGGER has tagged every proper noun in the sentence, this drastically reduces the amount of words to check.

Identifying Teams and Players mentioned

Identifying the correct player is not as simple as one might think because generally, names contain at least 2 different proper nouns (i.e. a first name and a surname). Furthermore, Twitter sources do not follow the same writing styles i.e. some sources will always mention the full name of a player while others will only mention the surname, it is not sufficient to only search for the first name or the surname as many different players share the same first name or surname. A successful method will have to search for a first name and surname and double check that both these refer to the same player. One way of achieving this would be to implement a recursive method, in which it searches for a first name match, and once a match is confirmed, the method would be called again to look for a surname match. For this to work, there are some things to consider:

- Must be able to distinguish between a first name and a surname
- Must be able to temporarily store the player, whose recursive search is looking for its surname after identifying its first name

A well-planned input file could solve all these issues. If the text file contained rows of tuples of which each tuple contained the first name and surname strictly in that order. An important consideration with using such a method to identify players is the efficiency of the method. Because the method is searching through a text file, an efficient search algorithm should be implemented to minimise the run time. By pre-sorting the text file, it is possible to implement a search algorithm. However, this poses another question of whether to sort by first name or surname which is not a straightforward decision to make. The decision is essentially about understanding what method would minimise the search time. For an effective method, it was important to predict every potential way a player can be referred to in a tweet. These are listed below in table 7:

What is mentioned in a tweet	Approach
First name and surname is mentioned following each other	This is the ideal situation. No more work is required (The current dataset contains no players sharing the same name – this will need to be considered in the future)
First name and surname is mentioned, but not following each other	It is expected for such cases to be extremely rare, therefore no work will be made. The script will not be able to identify the player mentioned in such cases
Only the first name of a player is mentioned	In such cases, the following process will take place:

	If list of matching first names has only 1 name and that player is playing for a team that has also been identified, then match Otherwise, no further actions will be taken to identify the player and tweet will be discarded
Only the surname of a player is mentioned	In such cases, the following process will take place in the order below: 1. *If list of matching surnames has only 1 name, then match the tweet to that player* 2. *If list of matching surnames has more than 1 name, and only one player in the matching list plays for a club also mentioned in the tweet, then match the tweet to that player* 3. *Otherwise, filter out the tweet as identifying a player requires the system to assume too much information that is not provided*
No recognized name is mentioned	Filter out the tweet

Table 7 : Approach to Identifying Players

Identifying the team mentioned is almost identical, and it will be possible to almost fully complete the task in the same recursive method, with just two small methods to compare the named entities with the list of players/team. One method will be to search for the player mentioned while the other will be to search for the team mentioned. The reason for this separation, is that matching the correct player/team with the player/team that the source is referring to in the tweet poses different problems and requires different actions to deal with these problems. Table 7 above mentioned the potential problems of identifying the player mentioned, table 8 below mentions the potential problems of identifying the team mentioned

What is mentioned in a tweet	**Approach**
Full team name	This is the ideal situation. No more work is required. This is under the assumption that specific words that proceed team names are not mentioned i.e. 'Football Club' or 'United' only when that is not the only way of distinguishing between different club for example – West Ham United can be shortened to West Ham as there are no other West Ham EPL teams, but 'United' is essential in distinguishing between the 2 Manchester clubs etc.)
Part of a team name (i.e. 'United' when referring to 'Manchester United')	It would be difficult to assign such a word to a specific team as 'United' could refer to many different teams, although the likelihood is that they are referring to the 'Bigger' teams, it cannot be taken for granted that they are. It would complicate the task of searching for links to confirm this, therefore in such a case, no team would be identified

Nickname of a team (i.e. 'Spurs' when referring to 'Tottenham')	To complicate matters, some sources might refer to the team by its nickname. In order to accommodate this, potential nicknames would have to be inputted via the text file of team, and an additional identifier to group different names referring to the same team.
No recognised team name is mentioned	This does not mean that there is no transfer news present, as a source might be claiming a player will leave without specifically mentioning the team the player is currently playing at, but the system will not be able to recognise these claims

Table 8 : Approach to identifying teams

Unfortunately, identifying what is required is not as simple as it sound as there are always complications. This project aims to identify and adapt different methods to counter these complications, however it is not feasible to do this for every potential complication. This will mean that the system will have some constraints and assumptions when processing tweets from different sources, which as a result will have an impact on the overall accuracy of the reliability scores. However, by managing the constraints and the assumptions, this impact will be minimal. Managing the assumptions and constraints essentially means that the assumptions will not equate to too much wrong information being admitted and the constraints do not equate to too much information being missed. For example, an assumption made by the system will be that if a source mentions a surname of a player and a club that the player is currently contracted to, it is assumed that the source was referring to that specific player, however it is not guaranteed that that is the case i.e. a source could mention a music artist with the same surname as a player playing for Chelsea, and the source mentions that the artist is due to play in Chelsea, London. This would mean that the assumption is incorrect, however it would be highly unlikely to come across such a case. Similarly, the constraints on players first name and surname not following each other resulting in non-identification means that some claims might be missed by the script. An example of such a case would be "Rooney set to leave. Wayne claims he would prefer Everton". The following tweet refers to the player "Wayne Rooney", however the constraint on the design of the script would mean that this claim is missed. However, it is reasonable to dismiss this case as being very rare, and sources do include the full name of a player in most cases.

Normalising Text

The next stage is to process the data into a normalised form, ready to be classified as either a relevant tweet or not. Section 2.5.2 goes into detail on what ML means. To apply an effective ML algorithm, the text will need to be split into meaningful features. Meaningful features constitutes of distinctive features between categories, such as specific verbs and nouns, however not proper nouns or specific symbols since such characters and words have equally the same probability of appearing in a tweet about football transfers or a tweet not about football transfers. Figure 10 on the next page shows the steps to be taken in this task:

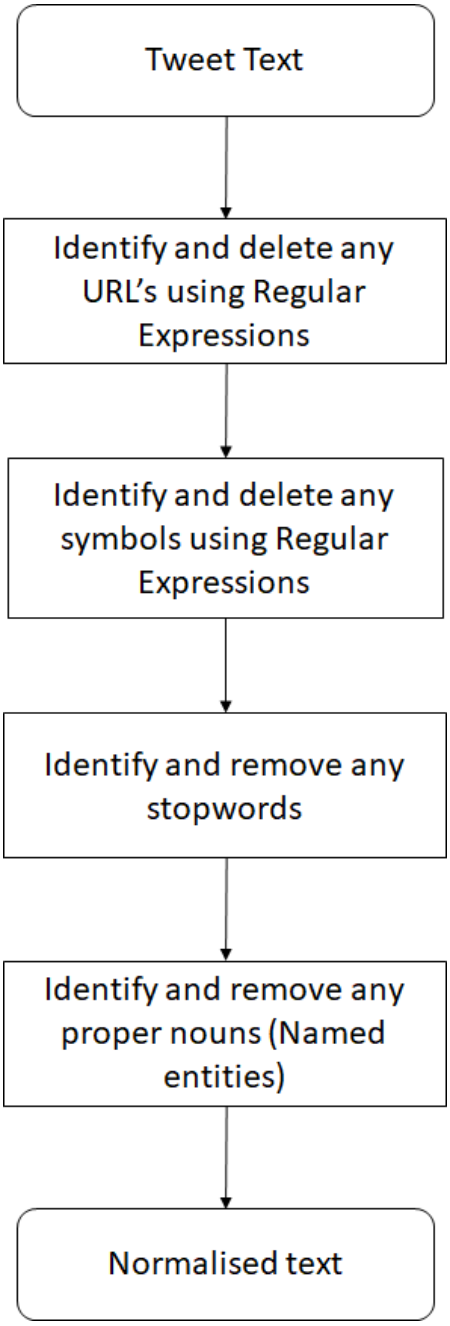

Figure 10 : Flowchart detailing steps for normalising twitter text

Once all these texts have been taken, a further task is required before the resulting text is ready to be passed on to the classifier that is based on the requirement of the classification method itself which is called from the NLTK module. Essentially this just means creating a tuple that consists of the list of all features identified by the classifier, that list will be looped and values will be changed to true if the feature occurs in the text, and false if not.

Once all these steps are completed, the dictionary will then be passed on as a parameter to the classify method of the classifier object.

Building the classifier

As mentioned before, the classifier that we will be using in this project is based on the Naïve Bayes algorithm. In terms of the actual construction of the classifier, there is very little to be done. The NLTK module takes care of all the technical aspects of the construction (The exact process has already been discussed in chapter 2). All that is required is to pass on the training set and the testing set to the build method. However, this is a crucial step as the training data is vital to the success of the classifier. Quality training data is vital. As discussed in the requirements, there were 2 possible ways of generating the dataset. The only potential problems with training data is that it could be biased, depending on the source of the data and whether some data in the main dataset is from the same source. In terms of implementing the two-different method, it is very straight forward:

1. Taking a random sample and manually categorising each tweet – This involves a simple function that will iterate through the entire data set, and take a sample every *n* tweet and write the tweet to a file, which is to be manually scanned through and labelled by a human. Of course, this means that there are some bias towards the tweets taken from the main dataset, but randomising our sample, this means that any possible bias is minimised.
2. Capturing tweets from a Twitter account where all tweets are known to include transfer news / non-transfer news – This is less work than the 1st method as it is automated to an extent. A simple script (Similar to the script used to capture the main dataset in section 3.1.2) can be utilised to capture the past 3200 tweets (Twitter API Limit) of a particular user. This can be used on a Twitter source that is known to only tweet transfer news and nothing else, similarly for a source that is known to only tweet football news not containing transfer news.

Unfortunately, due to the time limitation imposed on the project, it is unfeasible to do method 1 for such a time that there will be sufficient training data. It is estimated that hundreds of thousands of tweets will be in the main dataset, therefore a training set will need to include at least a thousand tweets. Manually categorising over a thousand tweets would be very time consuming.

Granted, using a training set that is not guaranteed to fully reflect the main dataset is risky as it makes the classifier prone to overfitting and the inability to generalise well to the unseen data, however it is predicted that the main features will be similar i.e. both will contain the keywords of 'transfer', 'signing' etc. To ensure that the performance of the classifier is satisfactory and reduce the risk of overfitting, the testing set will contain a random sample of the main dataset which will have been manually categorised, if the performance is deemed to be unsatisfactory, the alternative approach will have to considered (Unsatisfactory is a success rate in the region of 70% or less)

The classification process

The classify function itself is entirely made up of existing code from the NLTK module. Once the classifier is trained using the *trainclassifier* method, it is then a simple case of passing the tweet as a tuple of all words(as defined by the trainingset) which contains the Boolean

value to indicate whether that word is contained in the tweet or not in to a pre-defined *classify* function. The function then returns the classification of that tweet. Which then means that the requirement is completed and the task for this section of the project is completed.

3.3 Computing Trustworthy and Belief Values

Although the majority of the work is done in the data capture and NLP & ML section of the project, the only result of real value at the end of the project will be the reliability of each source – and as a result a value of belief for every claim made. This section of the project is all about dealing with the data we have available, and implementing a suitable algorithm that is able to compute values that represent trust and belief in sources and claims.

3.3.1 Requirements for computing trust and belief values

The requirements for the final output of the system are:

- A Trust(T) value for every source who has made more than n claims (n is an arbitrary number than is decided at runtime – this is to avoid outputting 100's of sources that had only made a single claim – difficult to have credible values if a source has only made a single claim)
- A Belief(B) value for every claim made by all the sources.

The system will output all T and B values for every source / claim at the end of every day of the Transfer Window up until the final iteration after all the data has been processed to add / subtract to the final Trust and Belief values.

In order to compute a trust and belief value, there are a number of variables required to be configured from the data as mentioned by (Pasternack & Roth, 2010). These are:

- Set of all sources
- Set of all claims
- Set of all claims a source makes
- Set of all claims regarding a player
- Set to store Trust values for all sources
- Set to store Belief values for all claims

Of course, these don't all have to be individual sets, they can be accessed from each other i.e. by using a Python dictionary to hold all the claims a source makes while setting the key to be the source will give us a set of all sources and claims a source make.

3.3.2 Deciding and adapting a suitable algorithm

The methods proposed in this project are taken from the paper Knowing What To Believe by J Pasternack & D Roth (Pasternack & Roth, 2010). The paper proposes several equations to compute Trust and Belief values given certain inputs that are derived from the previous works of many different researchers as mentioned in Chapter 2. The equations below are the ones to be used in the project, with the one giving the best result to be used to compute the main outcome of the project:

All equations use the input:

- S: Set of sources each asserting a set of claims C_s

$$T^i(s) = \sum_{c \in C_s} B^{i-1}(c) \quad B^i(c) = \sum_{s \in S_c} T^i(s)$$

Equation 4 : Sums

$$T^i(s) = \log |C_s| \cdot \frac{\sum_{c \in C_s} B^{i-1}(c)}{|C_s|}$$

Equation 5 uses the same formula as in equation 4 for calculating $B^i(s)$

Equation 5 : Average Log

$$B^i(c) = \sum_{s \in S_c} \frac{T^i(s)}{|C_s|} \cdot \frac{\mathcal{G}(H^i(c))}{\sum_{d \in M_c} \mathcal{G}(H^i(d))} \quad T^i(s) = \sum_{c \in C_s} B^{i-1}(c) \cdot \frac{T^{i-1}(s)}{|C_s| \cdot \sum_{r \in S_c} \frac{T^{i-1}(r)}{|C_r|}}$$

$G(x) = x^g$ where g is arbitrary

Equation 6 : Pooled Investment

Equation 4 is a simple but effective method of coming up with some sort of trust and belief score. It essentially sums all the belief values of every claim made by that source to get the trust value, similarly it sums up all the trust values of every source making that claim to get the belief value. On the first iteration, the belief value will always be 0.5

Equation 5 is just a simple improvement on equation 4. One easy way for a source with relatively low accuracy to get a high trust score in equation 4 is to simply make as many claims as possible. Equation 5 tackles this problem

Equation 6 is more complex, as the trust of each source is uniformly attributed to each claim it has made, and the belief in each claim grows exponentially according to an arbitrary power. The trust values for each source is calculated by summing the belief in all of its claims, weighted by the trust previously contributed to each claim.

2 Main adaptions are to be applied to these equations to provide better results in the context of this project:

- Divide all T values by the value of the largest T (The same also applied to B values) in order to have percentage values
- Have an artificial source with the maximum T value – apply all transfer tweets from official club accounts to this source – this allows the equations to make use of known

occurrences to better reflect the T values – Once a transfer is confirmed to have happened, the B value of all claims referring to the player in question is no longer useful.

4 Implementation

This section describes in detail the implementation steps taken to fulfil the requirements listed in the previous chapter by closely following the design that was proposed. All implementation was done in Python (The reasons as to why this was the case was listed in Chapter 2). This chapter also includes some basic testing on certain functions, this is to ensure they are functioning correctly before using the script on the main dataset to gather the final results. Every section within this chapter will provide detailed explanation of the implementation, explain key code snippets and provide details of the intermediate results achieved along with the problems that became evident during implementation.

4.1 Implementing a script to generate a suitable dataset

Although this section of the project is essentially a precursory task, it is vital that it becomes a success as without any suitable dataset, it would become very difficult to provide a proof of concept to suggest the feasibility of what this project is aiming to achieve. The actual implementation will compose of a standalone python script that will be separate to the main script used to process the data, however in the future both these steps will have to integrate to become a single step, where incoming data (tweets) are processed in real-time.

The requirements for this section were very straight forward as was the implementation since the script implemented heavily relies on external libraries which means less work is required to create and test a script. Algorithm 1 below shows the code for capturing data from tweets in real-time.

Algorithm 1 : Twitter_Streaming

```
1. Def Data_Capture(access_token, access_token_secret, consumer_key,
   consumer_secret, consumer_secret, eplplayers)
2.     L = StdOutListener()
3.     Auth = OAuthHandler)consumer_key, consumer_secret)
4.     Auth.set_access_token(access_token,access_token_secret)
5.     Stream = Stream(auth, 1)
6.     Stream.filter(track=eplplayers)
```

Algorithm 1 : Twitter Streaming

Algorithm 2 : Output Processing

```
1. class StdOutListener(StreamListener)
2.     def on_data(self, data)
3.         if 'retweeted_status' not in data
4.             print(data)
5.         return True
6.
7.     def on_error(self, status)
8.         print(status)
```

Algorithm 2 : Output Processing

As previously discussed, the parameters for the Data_Capture function are the authorisation credentials generated when registering for a Twitter developer account. The variable l is for the StdOutListener Object, which is essentially a buffer zone between the pulling of a tweet and the pushing it to the function. This class also provides some pre-processing – The filtration of retweets and some basic error reporting. This process can be seen in Algorithm 2 above. The next step in the algorithm is the creation of an OAuthHandler instance. Again, this step is briefly mentioned in Chapter 2 and is a requirement of the Tweepy Process. Similarly, the Stream instance is created according to Tweepy requirements. The filter method is a method which only returns tweets containing the key words which is listed as the parameter eplplayers. This is a text file containing a list of all the 772 Registered EPL players (As of December 2017).

The result of this process was a JSON output which contained huge amount of data and metadata on each individual tweet. Most of this data was not useful for the purposes of this project therefore a further script was required to convert it into a useful format – that was a CSV file containing 4 columns of data as was set out in the requirements. Figure 11 below shows the initial result of a captured tweet, before further processing .

{"created_at":"Tue Jan 02 18:03:55 +0000
2018","id":948253538051911680,"id_str":"948253538051911680","text":"BAHAHAHAHAHAHADNVO5JVJDJF
https:\/\/t.co\/EGwu2bdfPW","display_text_range":[0,25],"source":"\u003ca href=\"http:\/\/twitter.com\/download\/iphone\"
rel=\"nofollow\"\u003eTwitter for
iPhone\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to
_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":944599534780416000,"id_str":"944599534780416000","name":"Jimy
Danger","screen_name":"DangerJimy","location":"Mayo, Ireland","url":null,"description":"follow all ft mayo 4
sam","translator_type":"none","protected":false,"verified":false,"followers_count":69,"friends_count":174,"listed_count":1,"favourites_count":
215,"statuses_count":206,"created_at":"Sat Dec 23 16:04:13 +0000
2017","utc_offset":null,"time_zone":null,"geo_enabled":true,"lang":"en","contributors_enabled":false,"is_translator":false,"profile_background
_color":"F5F8FA","profile_background_image_url":"","profile_background_image_url_https":"","profile_background_tile":false,"profile_link_co
lor":"1DA1F2","profile_sidebar_border_color":"C0DEED","profile_sidebar_fill_color":"DDEEF6","profile_text_color":"333333","profile_use_back
ground_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/947274934325334016\/jypoesm7_normal.jpg","profile_im
age_url_https":"https:\/\/pbs.twimg.com\/profile_images\/947274934325334016\/jypoesm7_normal.jpg","profile_banner_url":"https:\/\/pbs.
twimg.com\/profile_banners\/944599534780416000\/1514682920","default_profile":true,"default_profile_image":false,"following":null,"follo
w_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,"quoted_status_id":947907789275389
52,"quoted_status_id_str":"947907789275389952","quoted_status":{"created_at":"Mon Jan 01 19:10:02 +0000
2018","id":947907789275389952,"id_str":"947907789275389952","text":"Ragnar Klavan. 94th minute. Limbs everywhere!
https:\/\/t.co\/JQzJGTzFCi","display_text_range":[0,45],"source":"\u003ca href=\"https:\/\/studio.twitter.com\" rel=\"nofollow\"\u003eMedia
Studio\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to
_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":136668622,"id_str":"136668622","name":"The Redmen
TV","screen_name":"TheRedmenTV","location":"Liverpool","url":"http:\/\/www.theredmentv.com","description":"Award winning Independent
Liverpool FC fan channel. Created by fans for fans. http:\/\/www.youtube.com\/TheRedmenTV Business Enquiries -
interact@theredmentv.com","translator_type":"regular","protected":false,"verified":true,"followers_count":120953,"friends_count":1813,"liste
d_count":959,"favourites_count":13322,"statuses_count":50288,"created_at":"Sat Apr 24 15:30:50 +0000
2010","utc_offset":0,"time_zone":"Casablanca","geo_enabled":false,"lang":"en","contributors_enabled":false,"is_translator":false,"profile_back
ground_color":"EB3636","profile_background_image_url":"http:\/\/pbs.twimg.com\/profile_background_images\/441230937213251584\/xJX
WjXKI.jpeg","profile_background_image_url_https":"https:\/\/pbs.twimg.com\/profile_background_images\/441230937213251584\/xJXWjXKI.
jpeg","profile_background_tile":false,"profile_link_color":"FF192C","profile_sidebar_border_color":"FFFFFF","profile_sidebar_fill_color":"E6E6E
6","profile_text_color":"333333","profile_use_background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/853587
133059276800\/Vk_RSZ6p_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/853587133059276800\/Vk_RSZ
6p_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/136668622\/1492504728","default_profile":false,"default_p
rofile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,
"is_quote_status":false,"quote_count":42,"reply_count":15,"retweet_count":82,"favorite_count":554,"entities":{"hashtags":[],"urls":[],"user_m
entions":[],"symbols":[],"media":[{"id":947894897071321089,"id_str":"947894897071321089","indices":[46,69],"media_url":"http:\/\/pbs.twi
mg.com\/media\/DSecokPWAAIpK2G.jpg","media_url_https":"https:\/\/pbs.twimg.com\/media\/DSecokPWAAIpK2G.jpg","url":"https:\/\/t.co\
/JQzJGTzFCi","display_url":"pic.twitter.com\/JQzJGTzFCi","expanded_url":"https:\/\/twitter.com\/TheRedmenTV\/status\/947907789275389952
\/video\/1","type":"photo","sizes":{"thumb":{"w":150,"h":150,"resize":"crop"},"medium":{"w":320,"h":180,"resize":"fit"},"large":{"w":320,"h":1
80,"resize":"fit"},"small":{"w":320,"h":180,"resize":"fit"}}}]},"extended_entities":{"media":[{"id":947894897071321089,"id_str":"947894897071
321089","indices":[46,69],"media_url":"http:\/\/pbs.twimg.com\/media\/DSecokPWAAIpK2G.jpg","media_url_https":"https:\/\/pbs.twimg.com
\/media\/DSecokPWAAIpK2G.jpg","url":"https:\/\/t.co\/JQzJGTzFCi","display_url":"pic.twitter.com\/JQzJGTzFCi","expanded_url":"https:\/\/twitt
er.com\/TheRedmenTV\/status\/947907789275389952\/video\/1","type":"video","sizes":{"thumb":{"w":150,"h":150,"resize":"crop"},"medium
":{"w":320,"h":180,"resize":"fit"},"large":{"w":320,"h":180,"resize":"fit"},"small":{"w":320,"h":180,"resize":"fit"}},"video_info":{"aspect_ratio":[1
6,9],"duration_millis":40400,"variants":[{"content_type":"application\/URL","url":"https:\/\/video.twimg.com\/amplify_video\/9478948970713
21089\/pl\/EqV8sJ7MSx1plIPg.m3u8"},{"bitrate":2176000,"content_type":"video\/mp4","url":"https:\/\/video.twimg.com\/amplify_video\/947
894897071321089\/vid\/1280x720\/RsbMaZJ5WVNNoGRN.mp4"},{"bitrate":832000,"content_type":"video\/mp4","url":"https:\/\/video.twim
g.com\/amplify_video\/947894897071321089\/vid\/640x360\/Zw_HWxr7oqf3mibz.mp4"},{"bitrate":320000,"content_type":"video\/mp4","url
":"https:\/\/video.twimg.com\/amplify_video\/947894897071321089\/vid\/320x180\/S5nHVYlLbQEIe1fX.mp4"}]}}]},"favorited":false,"retweete
d":false,"possibly_sensitive":false,"filter_level":"low","lang":"en"},"is_quote_status":true,"quote_count":0,"reply_count":0,"retweet_count":0,"
favorite_count":0,"entities":{"hashtags":[],"urls":[{"url":"https:\/\/t.co\/EGwu2bdfPW","expanded_url":"https:\/\/twitter.com\/theredmentv\/s
tatus\/947907789275389952","display_url":"twitter.com\/theredmentv\/st\u2026","indices":[26,49]}]},"user_mentions":[],"symbols":[]},"favorit
ed":false,"retweeted":false,"possibly_sensitive":false,"filter_level":"low","lang":"it","timestamp_ms":"1514916235827"}

Figure 11 : JSON Output for a single tweet

Evidently from the figure above, it was important that the JSON objects were further processed in order to make it simpler to manipulate and read the data, and to increase storage efficiency, as the large majority of the data on display here was not needed.

The implementation was very straightforward, with just basic JSON indexing and the utilisation of the CSV module for Python. Algorithm 3 below shows the steps taken to convert the lengthy JSON representation into a simple CSV row.

Algorithm 3 : JSON_to_CSV

```
1. def JSON_to_CSV(JSON_Input, CSV_Output)
2.     input = open(JSON_Input)
3.     with open(CSV_OUTPUT) as csvfile
4.         writer = csv.writer
5.         for line in input
6.             x = json.loads(line)
7.             if x['lang'] == 'en'
8.                 username = x['user']['screen_name]
9.             Writer.writerow([x['id'],x['created_at'],username,x['text']])
```

Algorithm 3 : JSON to CSV processing

The code above is self-explanatory. The input is the location of a text file that is the output of algorithm 1 (When Python output is directed towards a text file), while the output (2nd parameter) is the location where the output is to be saved. The index terms used are specific terms of the JSON object. This can be seen in figure 11 where there are matching indexes.

Intermediate Results

Soon after implementing the algorithms shown above, some problems arose and potential problems came forward which had to be addressed.

Firstly, there was a problem of the limitation on the number of keywords that can be applied to a filter. Upon testing the script, it was discovered that Twitter imposes a maximum number of keywords for the filter method which is 400, meaning that the algorithm was unable to take the list of 772 EPL players. Fortunately, it was possible to access a second developer account, and thus splitting the keyword list into 2 separate lists to be within the Twitter limit of 400 keywords.

Secondly, the scale of the task was not properly taken into account when designing the implementation. It is important to remember that the script will be running continuously without interruption for a period of 31 days (Covering the January Transfer Window) and the nature of the script – continuous HTTP requests that can at times be prone to timeouts and similar errors. Therefore, it was important to amend the algorithm to address these issues. Amending the algorithm involved basic error catching techniques and other methods

to better cooperate with Twitter's rate limitation strategies (explained later on). They included:

- Python's *Try Except* error catching
- Including the *Async* parameter in the stream method
- A timer of 60 seconds between new requests after any failures

Adding these to the code meant that the algorithm's capabilities to deal with any potential problems was greatly increased. Including a *Try* and *Except* error catch meant that the script would not crash should any error arise, along with the addition of the *Async* option as a parameter – which essentially notifies the Stream object not to terminate the connection unless the connection is closed. Should a stream return some trivial errors, the *Async* feature means that a new thread would be created and the script would continue to run on this new thread. Should any error occur and there is an issue in connecting to the Streaming API, it is important to limit the retries as Twitter enforces a rate limitation policy – this means that a client's access to the API is restricted for a number of seconds, however this timer is increased exponentially with every failed attempt to connect. Therefore, it is vital that attempts to reconnect are limited in order to avoid long periods of restricted access. Adding a timer for 60 seconds between every retry ensures that the number of attempts to reconnect is severely limited, and should the reason for the failed connection be a trivial problem such as an outage in connectivity, the timer would have allowed sufficient time for the problem to be potentially solved.

During the run time of the script, manual steps were taken to forcefully disconnect the stream after a number of days and reconnect again with the output directed towards a different text file. The reasoning behind this was that the stream was parsing thousands upon thousands of tweets a day, and it was better to have numerous smaller data files than one large file at the end of the process. This protected against the possibility of a file corruption (Not all data would be lost as there were numerous data files).

After the Transfer Window was shut, the script ceased to run. The next case was simply the process of merging all the data files together. After this was done, a text file containing all the tweets captured in its JSON form was the result. This file contained *178,933,549 KB* of raw data.

Once all the tweets were gathered and placed in a file, the next step was to iterate through every JSON entry in that file and extract only the data that was necessary, this was done using algorithm 3. The result is a CSV file which contains *86,810 KB* of data – a massive 99% decrease in size from the original JSON file. In terms of the number of relevant tweets captures, this was *432,018* tweets in total.

The final outcome of this stage of the process was very satisfactory, generating a dataset with plenty of tweets in order to test the rest of the project against.

4.2 Processing data using NLP Techniques and classifying text

The main project was split into three sections, this individual section also contained distinct parts, mainly the NLP and the ML parts. The first task was to implement a script that dealt with ambiguities and variances in the hundreds of thousands of different tweets captures from the first section of the project.

The underlying process of any NLP was discussed in chapter 2.5.2. Figure 9 in Chapter 3 shows a detailed flowchart of the process. The code used to implement is mostly taken from existing methods within the NLTK package. These are basic one-line functions which are used to:

- Tokenize – The tokenisation method used is to take individual words as tokens
- POS-Tag – The NLTK method which returns a tuple containing the word passed as parameter and the POS Tag abbreviation

These 2 functions are applied to the tweet before anything is done.

4.2.1 Identifying Player P & Team T

Once these 'pre-processing' steps are complete, the first real task was to identify any players and football teams mentioned in the tweet. Unfortunately, there weren't any suitable pre-existing methods available for this. Trying different Named Entity Recognisers including the Word Net Corpus didn't provide meaningful results as it did not recognize some player names or some club names. The only real alternative was a tedious string comparison method which processes strings, this posed a range of potential problems as discussed in chapter 3.2.2. Implementing a perfect algorithm, given the time constraint was not feasible in this case, as there are so many different ways of a source referring to players and teams in a tweet. An exhaustive method to cover all such different ways would be very time consuming to implement.

Algorithm 4 : Player and Team Identification

```
1. def findPosNamesTeams(tagged,players, teams)
2.    players_in_tweet = []
3.    potentialPlayers = []
4.    teamlist = []
5.    teams_in_tweet = []
6.    nextwordflag = False
7.    counter = 0
8.    for words in tagged
9.       if nextwordflag == True
10.        counter = counter + 1
11.        nextwordflag = False
12.        continue
13.     if words[1] == 'NNP'
14.        match = checkIfTeam(words[0], teams)
15.        if match
16.           for matches in match
17.               if (len(teams[matches[0]]) == 1)
18.                  teams_in_tweet.append([teams[matches[0]], counter])
19.                  teamlist.append("".join(teams[matches[0]]))
20.              elif (matches[1] == 0 and ((counter + 1) < len(tagged)))
21.                  proceedingwordmatch = checkIfTeam((tagged[counter +
   1])[0],teams)
22.                  if proceedingwordmatch
23.                     teammatch = set(matches[0] for matches in
   proceedingwordmatch if matches[1] == 1).intersection(set([matches[0] for
   matches in match if matches[1] == 0]))
24.                  if len(teammatch) == 1
25.                     pointer = teammatch.pop()
26.                     teams_in_tweet.append([teams[pointer],counter])
27.                     teamlist.append("".join(teams[pointer]))
28.                     nextwordflag = True
29.        else
30.           match = checkIfPlayer(words[0],players)
31.           if match
32.              if (((counter + 1) < len(tagged)) and (tagged[counter + 1][1]
   == 'NNP'))
33.                  proceedingwordmatch = checkIfPlayer((tagged[counter +
   1])[0],players)
34.                  if proceedingwordmatch
35.                        surnames = [matches[0] for matches in
   proceedingwordmatch if matches[1] == 1]
36.                        firstnames = [matches[0] for matches in match if
   matches[1] == 0]
37.                        playermatch = set(firstnames).intersection(
   set(surnames))
38.                        if len(playermatch) == 1
```

```
39.                          players_in_tweet.append(
   [players[(playermatch.pop())] ,counter])
40.                          nextwordflag = True
41.                else
42.                   surnames = [matches[0] for matches in match if
   matches[1] == 1]
43.                  firstnames = [matches[0] for matches in match if
   matches[1] == 0]
44.                  if len(surnames) == 1
45.                     players_in_tweet.append([players[
   (surnames[0])],counter])
46.                  elif len(surnames) > 1
47.                     for surname in surnames
                           potentialPlayers.append(players[surname])
48.                  elif len(firstnames) == 1
49.                     potentialPlayers.append(players[firstnames[0]])
50.        counter += 1;
51.
52.     for potentialP in potentialPlayers
53.        if len(potentialP) == 1
54.           if (PlaysFor(potentialP[0])) in teamlist
55.              players_in_tweet.append([potentialP,counter])
56.           else
57.              if (PlaysFor(potentialP[0] + " " + potentialP[1])) in
   teamlist
58.                 players_in_tweet.append([potentialP,counter])
59.
60.     return players_in_tweet,teams_in_tweet
```

Algorithm 4 : Player and Team identification

Algorithm 4 above is one of the most important piece of code in the system. It is responsible for identifying and extracting the P and T variables in a tweet - that is the player(s) mentioned and the team(s) mentioned. Although the algorithm at first glance looks complex, it is essentially a string manipulation function. The parameters to the functions are:

- Tagged: The input tweet which has been tokenized and speech tagged
- Players: List of all official EPL players
- Teams: List of all 20 EPL teams

The first few lines in algorithm 4 are just variable initializations. The first task of the code is to decide whether the token (individual word in a tweet) is a Proper Noun. As part-of-speech tagging has previously occurred, this method simply compared the tag passed on as a parameter for every word. If the token is not a 'NNP' (Abbreviation for a Proper Noun) then the method continues to the next token in the for loop. This method of deciding whether to search for a matching player or club name is a simple and effective way, however it does depend on the sources following the formal linguistic rule of starting every name with a capital letter. If a player or team name does not begin with a capital letter, the token might not get tagged as a NNP. This is a constraint on the system in that it does not

deal well with sources not following conventional writing standards, but it is reasonable to expect every source to begin names with capital letters, even in the world of social media. The next part of the algorithm is split into two conditional if else statements, where the first statement will check if the token can be used to identify a team, and if this returns false, the else statement will check if the token can be used to identify a player. To compare the token against a player / team, algorithm 4 will call another search function which essentially consists of a few lines of code that opens a .csv file containing a list of all the registered EPL players / teams and then search for the player / team in question, and returns a list of all the matched player(s) / team(s) if there are any.

The search method will return a list of tuples where each tuple contains 2 entries:

- The index of the match in the file of teams / players passed as a parameter to the function
- Its position in the line where there is a match for example – 'Manchester' in 'Manchester City' would return 0 (Zero Indexed) as its position in the line is 0 whereas 'City' would return 1

The same example for a Player match if the token passed was 'Romelu' and the only matching entry in the players file is 'Romelu Lukaku' at index 556 (An example) would be – (556,0)

Once algorithm 5 has been run and has returned the list of tuples corresponding to every matched player / team, algorithm 4(findPosNames Function) will then either:

- Terminate (After a search for a player and team has been made) - If no match was returned – PotentialPlayers list will always be empty in this case therefore the entire function will go to line 57 and return an empty list
- Proceed to iterate through all matches

Figure 12 on the next page shows an illustration to help explain what this part of the algorithm does to identify the team T.

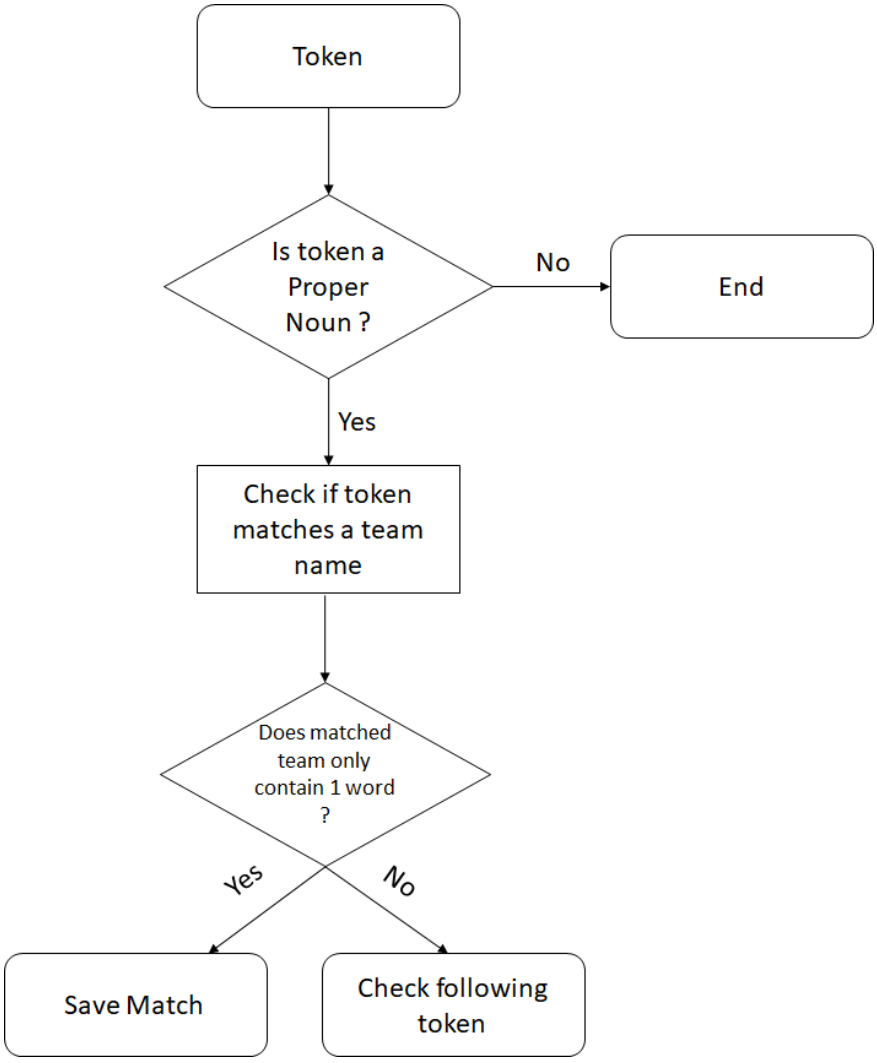

Figure 12 : Illustration of identifying T

The process for searching for the player or team once a team or player has been matched to a token is very similar to each other. The method is to iterate through every matched tuple that was returned from the previous function, and during each iteration, check if the matched team reference team name only contains one word (When working on a matched team not player) i.e. 'Arsenal' contains one word whereas 'Manchester City' contains two. If that is the case then the system can be reasonably certain that that team is indeed what the source is referring to and the algorithm will then add that team to a list of identified 'T' values. If the matched team reference contains a match on a team which consists of more than one word i.e. 'Manchester United', then the algorithm will then proceed to check that the current token it's in the process of checking within the iteration is not the last token in the tweet via the use of a counter variable and a quick length check on the list of all tokens – This is simply an error detection technique to avoid calling the method with an index of the tagged variable that is out of bounds. It will also check that the match is also referring to the first part of a team name i.e. its index in the line of the text file is 0 since the requirement noted down in 3.2.2 table 8, noted that the system would not be able to deal with a source

only quoting a part of a team name such as ‘United’ as opposed to ‘Manchester United’, therefore if the first match doesn’t contain a word in index 0, there is no way for the system to be certain which team the source if referring to. If there are indeed more tokens in the tweet, then the algorithm will proceed to call the team search function again, but with the next token in the tweet. The purpose of calling the same function again with the proceeding token is to confirm the exact team the source is referring to. After the function returns all the matches from the proceeding token, the system must check if there are any matches corresponding to the same team when the first token was passed to the function. This is done on line 23 by creating a set of the team indexes of all matches referring to the first word (index 0) and intersecting this set with a set of the team index of all matches referring to the second word (index 1). The result of this intersection is the index of the team that was referred to in both tokens passed on to the search function.

Once a list of team(s) T has been identified, the function then proceeds to do a length check on the list to decide whether no teams, a single team or multiple teams have been identified in the tweet. What will happen in each case is described below

- 0 Teams: No team has been identified, given the constraint on the system that it only deals with Premier League Teams, this case is expected to occur regularly, however in order to allow the system to deal with Premier League Players moving from a Premier League Team to a different league, the system allows for this to happen, and the value assigned to T will be the String: ‘other’
- 1 Team: A single team identified, this will only be passed on to the next processing stage in the following cases:
    - If only a single player is mentioned and the team that the player is currently playing for is not the same Team as has been identified here
    - If more than one player is mentioned and neither of those players are playing for that Team identified
- More than one team: Like the previous case, the system will remove any teams identified if it has also identified a player currently playing for the same team. After this has been done and if there are still more than one team mentioned, the tweet will be discarded. Although the usage of lists here does allow for the system to identify more than one team, this was implemented as a means to allow the system to improve and expand in the future. At present, when there is more than one team, the system will not be able to process it. This is a clear limitation of the system at present.

Note that the processes listed above do not occur in Algorithm 4. Algorithm 4 simply returns the list of all teams identified according to the requirements set out it 3.2.2 table 8, processing the cases listed occurs later in a different function. Every time a proceeding token is also checked, a Boolean variable called ‘NextWordFlag’ is set to True, the purpose of this is to avoid doing the same steps for word just checked in the following iteration, at the start of every iteration a quick check on the status of the flag is made and if it is True,

the algorithm resets the flag and skips that iteration. This process can be seen on line 28 (Setting of the flag) and on line 9 (Checking status of the flag). In terms of the second condition where the algorithm aims to identify all the players mentioned in the tweet, as previously said it is almost identical. Line 32 and 33 checks the following token for any possibility that they could be referring to the same player (i.e. first token mentioning the first name and the following mentioning the surname). The separate search function then compares matches to actual player names taken from the input file returns the index of the player in the file, and the position of the matched word within the full name – identical to the corresponding function for the team search. Using this index, the algorithm can differentiate between the first names and surnames matched and then intersect the sets of the indexes of both first name and surnames in the input file to check for matches referring to the same player – this happens in the lines 34 – 36. If no matching token occurs after the initial match, the algorithm only has one name to go off. The strategy here was discussed and laid out in table 7 chapter 3.2.2. The algorithm's usage of an additional variable 'PotentialPlayers' here is too check whether the player's mentioned here also plays for the team mentioned in order to decide whether to identify the player matched. The actual process and justification is described in plain English in Table 7.

Intermediate Results

The purpose of the functions described previously was to identify the P(player) and T(team) identifiers. It was stated clearly from the beginning that the methods proposed and designed would have numerous limitations and made some assumptions. The initial results witnessed from the functions are satisfactory for the purposes of this project. Most tweets do indeed mention the first name and surname of a player when referring to them, and the only time a player will fail to be identified from a tweet is when this does not happen. The only issue discovered after implementing the method was that the function did not take into consideration players who are often referred to using more than 3 tokens (individual words) such as "David De Gea" for example, although this is not a major issue as the function simply truncated the output to "David De", and as long as the input file is also truncated this is not an issue at all. Even so, it would be easy to modify the function to iterate through a list of all the tokens in a name should it be necessary to fix this issue in the future. Given the time constraint of the project, there was very little benefit in adapting the code to take into account such cases.

4.2.2 Text Normalization

Having identified the Team T and the Player P and knowing the User U, it is not enough to simply assume that the tweet is mentioning a potential transfer, the method proposed is a ML Classification based on certain features outlined in the requirements in chapter 3. The first task is to normalize the tweet in order to remove certain features that does not depend on the context of the tweet – this is explained in greater detail in the requirements and design(Chapter 3.2.1 and 3.2.2) The normalisation script produced is seen below as algorithm 5

Algorithm 5 : Normalize

```
1. def normalize(text, lemmatizer)
2.     NoURL = re.sub(r'http\S+', '', " ".join(text))
3.     alphanumeric = re.sub(r'[^a-zA-Z0-9 ]+', '', NoURL)
4.     wordslist = []
5.     tagged = nltk.pos_tag(word_tokenize(alphanumeric))
6.     for word, tag in tagged
7.         wordslist.append(lemmatizer.lemmatize(word,
   convert_tag_format(tag)),tag])
8.     words = find_words(wordslist, 2)
9.     return words
```

Algorithm 5 : Normalize

The parameters are simply the text(tweet) itself and a lemmatizer object which is imported from the NLTK.Stem package. The object takes care of lemmatizing every token. However, there was a problem of different packages using different format to represent part of speech tags, therefore a further function which simply converted between the format used by NLTK (Current format) in to the format used by the WordNetLemmatizer used in the algorithm – this can be seen in line 7. Before this, as was defined in the requirements in figure 10, line 2 uses regular expressions to eliminate any URL from the tweet while line 3 uses regular expressions to get rid of any symbol that is not a number or a letter. The final requirement laid out in figure 10 was the removal of stopwords and any named entities. This is done by calling a separate function which basically iterates through all the tokens and makes comparisons against a list of stopwords and checks the tags of every token, removing the token from the list if either checks come back positive. Once the normalisation function has ran, it returns a version of the tweet that is stripped to its bare features. Table 9 below shows an example of what the function returned when example tweets were passed to the function.

Report: #Southampton want #Liverpool striker Daniel Sturridge on loan | Sports Mole #SaintsFC https://t.co/u4VOJkHq6E	want, striker, loan
West Ham United are closing in on the signing of Swansea City defender Alfie Mawson, according to sources close to… https://t.co/OHqOhYZ9ZF	close, signing, defender, accord, source, close
Now Chelsea eye transfer of Arsenal's Olivier Giroud as their hunt for a target-man continues: Striker's propo.. https://t.co/VBmXTkhqu9	Now, eye, transfer, hunt, targetman, continues
Chelsea sub on Willian and Pedro for Hazard and Fabregas, because they are only allowed to have two attacking player… https://t.co/nGaWiXQ67r	sub, allow, two, attack, player
Jake Livermore to escape FA punishment after confronting West Ham fan #WestHamFC https://t.co/BRXSZxN5Ar	escape, punishment, confront, fan

Table 9 : Example of the normalisation process output

These are only 5 examples out of over 400,000, but even so, it intuitively shows how this works as evidently there will be a pattern of features when a source makes a claim

regarding a transfer as opposed to a more randomised selection of words when referring to a different context.

4.2.3 ML Classification implementation

Having stripped down a tweet into a normalised form, the next problem was whether the tweet is indeed claiming a transfer is about to occur or not. To classify whether that is the case or not, a classifier needed to be trained. Training the classifier firstly involved passing on a training dataset. Two different ways of doing this was proposed in chapter 3.2.2. The first method, which was taking a random sample and manually categorising each tweet, was done first, if the result were satisfactory then the second method would not be required but it is predicted that this would not be the case for the reasons given in the requirements (3.2.2). Taking a sample of the dataset was a straightforward process, all that was required was to iterate through the dataset file, and for every row, compare the value of the row counter modulo n, against 0, if there is a match, then the tweet from that row is copied over to the output file (n is a value to be selected, where every n rows, that row is taken as a sample). The output of this method was then written to a CSV file containing a single column where each row was a tweet taken from the main dataset. The next step was simply a process of manually deciding whether the tweet was referring to a transfer or not, and writing one of two labels in the proceeding column – *transfernews* or *normalnews*. This was a time-consuming process; therefore, it was not feasible to do this to a sample of thousands of tweets. It was decided to create a dataset of 500 manually categorised tweets from a sample of the main dataset, however a portion of this dataset had to also be used as the testing dataset. The next steps before training the classifier was to format the training and testing dataset into a suitable input for the trainclassifier method of the NLTK NaiveBayes object. This means generating a feature set for every token(tweet) which means completing the following steps:

1. Creating a list of every word in every token in the dataset
2. Limit the list of every word into a suitable number
3. Assign every possible word, a True or False value depending on whether or not it appears in the tweet in question
4. Create a list of tuples for every tweet, where each tweet contains the output of the previous step and the category of that tweet

Figure 13 below shows a simplified example scenario and the output at each stage. In the example provided, the list of all words was limited to only three. The purposes of limiting this list is to make the function run much quicker – given that there are over 400,000 different tweets, the total number of all words to be discovered is going to be large and processing so much features will slow down the process too much.

Input

```
Tweet 1 : [transfer, move, completed, transfer, contract, deal], Category : transfernews
Tweet 2 : [move, stay, happy, extend, contract], Category : normalnews
```

Output

```
Stage 1 Output : All_Words = [move, completed, transfer, deal, stay, happy, extend, contract]
Stage 2 Output : All_Words = [contract, transfer, move]
Stage 3 Output : Tweet 1 = features('contract' : True, 'transfer' : True, 'Move' : True)
                 Tweet 2 = features('contract' : True, 'transfer' : False, 'Move' : True)
Stage 4 Output : Tweet 1 = {'contract' : True, 'transfer' : True, 'Move' : True}, transfernews)
                 Tweet 2 = {'contract' : True, 'transfer' : False, 'Move' : True}, normalnews)
```

Figure 13 : Formatting Training Data

There were a couple of functions implemented to make this happen, the main function – seen below as algorithm 6, is the function that gets the initial input (individual tweet) and returns the final output. It also gets the list of all words to look for, which is computed in a separate function, seen below as algorithm 7.

Algorithm 6 : Set Training Data

```
1. def settrainingdata(trainingdatafile, documents)
2.     trainingdata = open(trainingdatafile, encoding="utf-8", errors="ignore")
3.     trainingdatareader = csv.reader(trainingdata)
4.     for row in trainingdatareader
5.         documents.append((cl.normalize(row[0], lemmatizer), row[1]))
6.     random.shuffle(documents)
7.     Trainingfeatureset = [(find_features(text, word_features), category) for
   (text, category) in documents]
8.     Return [documents, trainingfeatureset]
```

Algorithm 6 : Set Training Data

In algorithm 6 above, the input parameters are the file where the tweets to be used as training data are located, while documents is just an empty list. The function is straightforward – The first few lines deal with opening the file in the correct format, then iterating through every tweet, normalising according to the NLP normalisation function (Algorithm 5). The purpose of shuffling the order of the tweets in line 6 is because splitting the dataset in to training and testing set needs to be random and not in any predictable order. Finally, it calls the featureset function which performs the required processes to get to stage 4. Along with the normalised text and category, the find_features function also requires a second parameter which is shown to be a variable - word_features on line 7. This variable is the output gained from the second step (Limited list of every word found in the training set). Algorithm 7 below shows the one-line function taken to convert a list of all words found in the training set (Which was retrieved via a single iteration of all words in

every tweet and added to a list) and output a list of n of the most important words in the training set

Algorithm 7 : Get Word Features

```
1. def getWord_Features(all_words, noofwords)
2.     word_features = [w[0] for w in sorted(all_words.items(), key=lambda
   k_v:k_v[1], reverse=True)[:noofwords]
```

Algorithm 7 : Get Word Features

The all_words input is the result of applying the *NLTK.FreqDist* method on the list of all words, which returns a dictionary where each key is one of the words in the list and the entry is the amount of times that word has appeared. This line sorts the dictionary by its entries (items) not the keys, in reverse order i.e. with the word that has appeared most starting at position 0. The sort function outputs a list of just the word itself not the counter (shown as w[0]) and finally the output is limited by the noofwords variable which is the input parameter.

To get to the final output stage (Stage 4), the find_features function is called, which simply consists of iterating through the tweet passed to it as an input and checking whether every individual token in that tweet is in the word_features list (output of algorithm 7) and denoting a True / False value whether it is contained in the list or not.

Results

During the implementation, a sample of nearly 500 tweets were taken (*489 Tweets*) This number is very small, just over 0.11% of the total tweets, however given the time available, generating and manually categorising a larger sample was not feasible. The result of using this sample as the training data and the testing data is shown below in table 10. It shows 3 tests, where a different percentage of the data was used to train and test the classifier (The higher the training percentage, the greater the number of tweets used to train the classifier at the expense of less data to test it on)

Training	Testing	Accuracy
50%	50%	61%
75%	25%	62%
90%	10%	61%

Table 10 : Classifying using a sample data set

With very similar results for the three tests, and the result being poor, it was clear that the classifier needed much more data to train with. Therefore, it was decided to use an alternative approach of using a small number of Twitter accounts that are known to only publish tweets of a specific category i.e. only transfer news or no transfer news at all, capture some of these tweets and then use these as the training tweets. The testing dataset would remain the same as this is an accurate representation of the main dataset. Using a very similar script to the one used in the first phase (generating suitable dataset Chapter

4.1) of this project to capture past tweets from four different Twitter accounts. Because of privacy concerns, the different accounts used cannot be named, but they were

- 2 accounts solely used to publish transfer news.
- 2 accounts solely used to publish statistics.

Using two different accounts for both categories meant there was less bias on a certain type of writing style that might have been employed by one of the sources. Again, applying the same processes as did with the previous training set, a total of *3179 Tweets* which is a *550%* increase in the training data available for the classifier to use. The result of training the classifier using this training dataset resulted in an accuracy of *74%* (Tested on the sample dataset previously generated, but using the entire dataset and not splitting it into training/testing set). An increase in accuracy from 62% to 74% was a great result, and a NB classifier with an accuracy of 74% is good and satisfactory for this project.

Combining the methods here along with the methods implemented previously (Chapter 4.2.1), it was then possible to iterate through the entire dataset, and from every tweet extract the necessary information from the tweet result in one of two possible outcomes:

1. Tweet does not contain transfer information or it contains too much information for the system to process
2. A Formal Logic statement of the form User ***U*** Claims Player ***P*** will join Team ***T***

Iterating through the entire dataset gave a final result of *97788* discovered claims. Which meant that *334,230* tweets had been discarded. A further filtration step was then applied to these claims in the next step (See next section).

4.3 Computing Trust & Belief from set of claims

This section of the process required firstly for the creation of different sets as defined by (Pasternack & Roth, 2010), before implementing the algorithm of choice.

4.3.1 Creating the necessary data structures for computation of Trust & Belief

As was defined in the requirements in chapter 3.3.1, before applying any algorithm to compute the values, the list of all claims had to be processed into different data structures. The different data structures required and their functions are:

- Predictions: Python dictionary where key is the source's ID and entries are lists of claims and metadata about the claim.
- C: Python dictionary where the key is a player's name, the entries are tuples consisting of a team where the player is claimed he will move to, along with a list of all the sources making that claims.
- T: A dictionary to store the trust values of all claims.
- B: A dictionary to store the belief values of all claims.

These data structures grow as the system iterates through all the claims made. Both the algorithms implemented are listed below. Algorithm 8 show's how details about the source is added, along with the claims it made, while Algorithm 9 focuses on the claim itself.

Algorithm 8 : Fill Source Entries

```
1. def addPrediction(dictionary, tweetdata, player, team, T, initialT)
2.     userid = tweetdata[2]
3.     date = tweetdata[0]
4.     datagathered = [parser.parse(date), player, team]
5.     if not (userid in dictionary)
6.        Dictionary[userid] = [datagathered]
7.        T[userid] = [initialT, initialT]
8.     else
9.        if datagathered[1] not in [claims[1] for claims in dictionary[userid]
10.          dictionary[userid].append(datagathered)
```

Algorithm 8 : Fill Source Entries

In algorithm 8 above, the *tweetdata* parameter consists of the list which is a result of the classification process, its format is *[date, time, userid, player, team]*. Line 4 sets the format that is to be entered as the claim in the prediction data structure. Line 5 and 8 are the conditional branches, with the condition being whether a source has already been registered (i.e. made a claim). If it has, it is a simple case of adding the claim and setting its initial trust value. If not, the function checks whether the source has previously made a claim about that player(line 9) and if it hasn't, the claim is appended to the list of all claims the source has made.

Algorithm 9 : Fill Claim Entries

```
1. def addClaim(C, source, player, team, B, initialB)
2.     if not player in C
3.        C[player] = {team : [source]}
4.        B[player, team] = initialB
5.     else
6.        if team in C[player]
7.           if source not in C[player][team]
8.              C[player][team].append(source)
9.        Else
10.          C[player][team] = [source]
11.          B[player,team] = initialB
```

Algorithm 9 : Fill Claim Entries

Algorithm 9 is also very similar. Initially there are two conditional statements (line 2 & 5) that checks whether a claim about the player in question has already been made, if not, it is simply a case of adding the player to the C data structure and an initial belief value for that claim. The C data structure is quite complex as it contains a dictionary within a dictionary - the format is that the player name is the key for the outer dictionary, and the entry for that

key is another dictionary, where the team that it is claimed the player will move to is the key and the items within that dictionary are a list of sources making the same claim. If a previous claim about that player has been made, the function then proceeds to check whether any claim has been made regarding a transfer to the same team (line 6) and if so, whether the claim has not previously been made by this source (line 7)

Once the dataset of claims has been iterated through and the above two algorithms has been run on all the claims, we are then able to retrieve a lot of stats about the claims that we have retrieved over the course of the January transfer window, these are seen below.

Set of Claims and Sources

Initially, after the classification process we had a dataset of around *97788* claims, made by *16999* different sources, which meant that an average source made around *5.75* claims. However, implementing algorithms 8 and 9 meant that repeat claims were not recognized along with a further process to filter out claims made on the same day or later than the day a transfer was officially completed (More on this later in chapter4.3.2), this finally meant that a total of *38226* 'valid' claims were made (Valid according to the definition set out in this specific project) While the average of 5.75 might seem reasonable at first glance, when computing the average after filtering out 'invalid claims', the average source only made *2.25* claims. Of course, it would be very difficult to accurately calculate trust value if on average sources only made 2 claims each, however, it is important to consider the number of sources that has only made a single claim – *12593* which meant that *25,663* claims were made between *4,406* sources meaning that the average source making more than one claim, will make *5.8* claims – which is a rather small number disappointingly, but more analysis on this is done in chapter 5. The reason for such a large number of sources only making one claim is likely to be that they are normal Twitter users who are not in the business of publishing football news but rather speculating football fans making their thoughts known to the Twitter community base.

4.3.2 Implementing an algorithm to calculate Trust and Belief values

As the project was severely constrained given the time period allotted, the algorithm to compute the all-important trust and belief values would have to be taken from existing research with some adaption as there was no time to propose completely new methods. Three different methods were implemented as explained in the approach (3.3.2 – equation 4,5 & 6). The code itself was identical to the equations with the two minor adaptions implemented as proposed:

- Adding an artificial source called 'Official' who's Trust value is hard coded to always achieve the maximum along with the belief in its claims also hard coded to always achieve the maximum belief
- Every Trust and Belief value is normalised to lie between 0 and 1 by maximum every value with the maximum value achieved in the iteration

Before any computation, the data is again filtered to ensure no claim has been made regarding a player after that player has officially transferred to another club. Because the accuracy of the NB classifier meant that some official tweets confirming a transfer was likely to be missed, this data was manually accumulated at the end of the transfer window – which meant that no time data was available for official transfers, therefore it was decided that any claim made regarding a transfer on the same day that the transfer actually occurred would be deemed invalid. This is not a poor constraint when you think about it – as it is not difficult to predict a transfer a couple of hours before it happens, most of the sources making these claims this late would almost certainly be going off other reporters making the same claim – this perhaps explains why there is such a reduction in the number of valid claims after applying this constraint. Implementing this case was simply a case of comparting dates whenever a claim was made about a player within a list of officially confirmed transfers. This list was inputted to the algorithm as a CSV file.

After implementing the Sums and Average Log algorithms, it was decided that their performance was satisfactory for the purposes of this project. Pooled Investment was also coded, but during initial tests, it was decided that there was no real purpose as the runtime of the algorithm was much slower than for the other two methods, and while it was possible to compare results at the end, there was no way of knowing which result better represented the situation.

The algorithms used required a specific amount of iterations to get a stable result, to determine the sufficient number of iteration required, a test was devised where:

- A random source and a random claim was selected
- Scores of these would be compared after a set amount of iterations were ran
- Once the difference in the score was small enough, the system would always iterate the process that number of times

The table below show the results: (The test used the entire dataset to achieve these results)

Number of iterations	Source **S**		Claim **B**	
	Sums	Average Log	Sums	Average Log
1	0.1276595	0.5469394	0.8440727	0.8044951
2	0.1897646	0.6003407	0.7689913	0.7437362
3	0.1992521	0.5742397	0.7433408	0.7329839
4	0.2014394	0.5694816	0.7419533	0.7312664
5	0.2015292	0.5693853	0.7416024	0.7310246
10	0.2015334	0.5693853	0.7414875	0.7310306

Table 11 : Experimenting with convergence

Based from the results shown in Table 11, it was decided that the algorithm should iterate 5 times to get a satisfactory result.

Finally, the system is set up to allow the user to input a date, to which the system will then return a trust value for all sources and the belief values for all claims at that date. The user will then be able to query certain sources and claims and sort them as required. This is done via a command-line interface where the user enters text based commands.

5 Result and Evaluation

In the previous chapter, a brief mention about the intermediary results of implementing each stage was discussed. In this chapter, these results will be evaluated in more detail, while the overall results of the project – The trust values of different sources and belief values of different claims will be shown and evaluated. Figure 14 below summarises the initial dataset retrieved and how it was reduced in size over many steps.

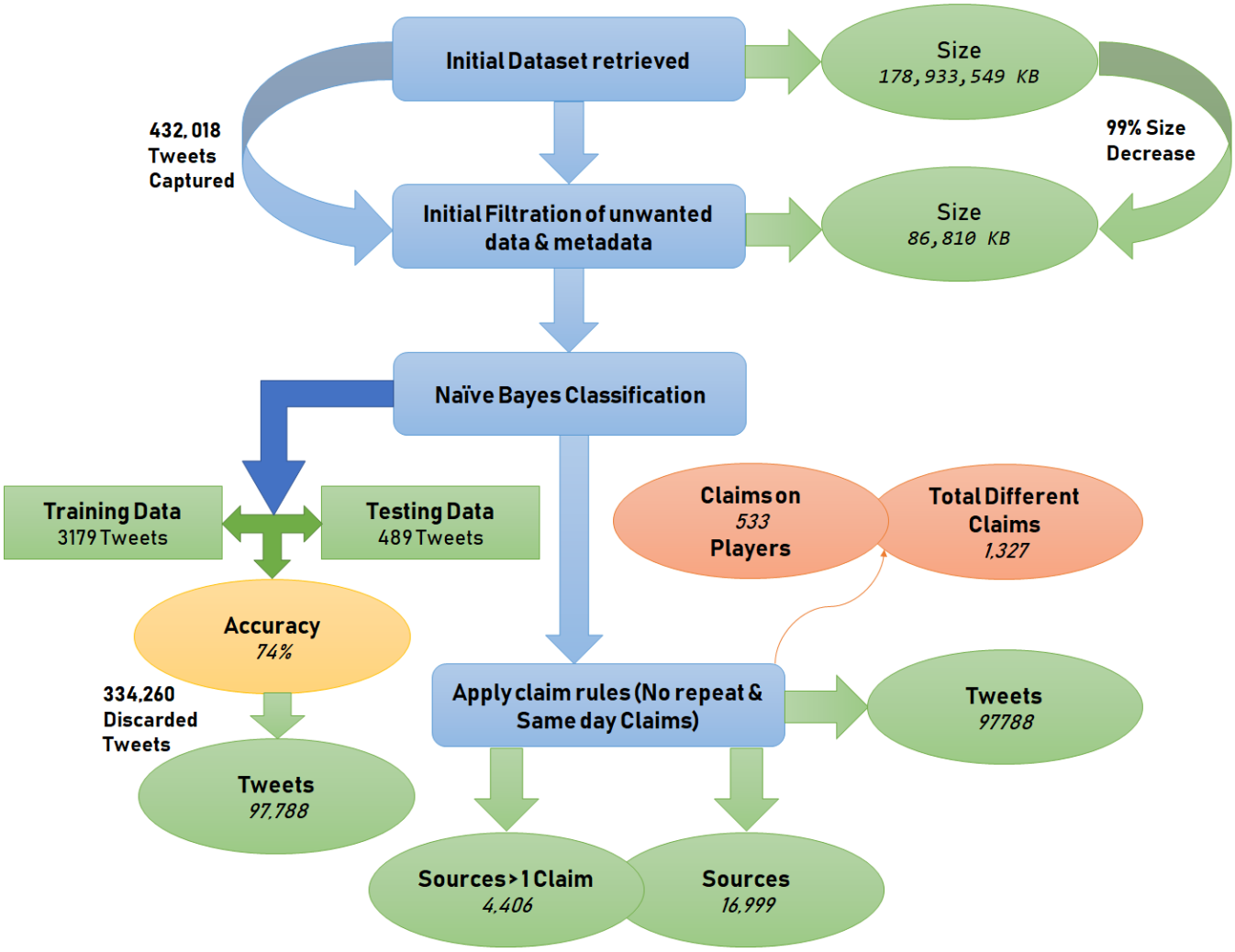

Figure 14 : The dataset in numbers

The previous chapter also showed the experimental results of trying out different numbers of iterations to apply for both algorithms to get a sufficient convergent. What was interesting from these experimental tests (table 11) was the large difference in trust values obtained from the different algorithms. In this chapter, the difference between the two algorithms used will be evaluated along with what the final trust and belief values taught us regarding what types of sources are most reliable.

5.1 Comparison of Sums and Average Log

To understand why there is such a large difference, it is important to consider what exactly are the differences between both algorithms, which is discussed in chapter 3.3.4. The Average Log minimises the advantage gained by making many claims (That are not necessarily likely to happen in reality). Because the algorithm used does not 'punish' sources for claims that have been known to be incorrect after they have made that claim, a source can easily obtain a better score by simply making as many claims as possible in the Sums algorithm. Average Log minimises this issue by dividing the score obtained with the number of claims made. A possible reason for such a difference in the score between both algorithms is that the Average Log algorithm will give a trust score of 0 to any source that has only made a single claim because Log (1) = 0. This means that 12,593 sources got a score of 0 which could easily explain the massive difference in the output of both algorithms.

Table 12 below compares how different sources performed when running different algorithms (The identification of every source is hidden to comply with ethical policies of the school) The test performed ignored the artificial 'Official' Source that is added. It also used the entire dataset.

Source	Sums			Average Log		
	Rank	Score	Claims Made	Rank	Score	Claims Made
S_1	1	0.95	66	45	0.79	66
S_2	273	0.27	8	1	1.0	8
S_3	48	0.66	52	396	0.61	52

Table 12 : Comparison of Sums and Average Log

Table 12 again confirms the large difference between trust values outputted by both algorithms. The test performed clearly indicates that there is a link between the number of claims made and the value received as one would expect looking at the scoring algorithm. By comparing the ranking and scores of S_3 it also demonstrates that the Average Log algorithm produces much higher scoring sources, since S_3 is ranked much lower by the Average Log even though the difference in score isn't much lower. Because the fact that S_2 was able to be ranked as the most trustworthy source despite making 8 claims – significantly less than most of the top sources clearly demonstrates that Average Log's goal of minimising the advantage from making many claims, clearly works. However, it could also be said that a source making only 8 claims, shouldn't be ranked so highly given that it has only made a small number of claims. It is a balancing act, and unfortunately there are no other sets of result to compare with and decide which algorithm gives the better results. Given the fact that the methods used to extract claims is far from perfect – Some claims might be missed and some tweets can be mistakenly picked up as claims, there shouldn't be any advantage handed to a source solely based on the amount of claims they make. Therefore, the remainder of the evaluation will be made on values gathered from using the Average Log algorithm.

5.2 Top Sources

The key outputs of the system implemented for this project is the list of sources and claims along with their trust and belief values. The results gives a chance for analysing the sources that were deemed the most trustworthy by the algorithm implemented. Table 13 below shows the top 10 sources identified alongside the type of source they are. The possibilities are:

- Freelance Journalist: A Twitter account belonging to an individual that does not claim to represent any media company
- Algorithm: An algorithm that published trending news
- Story Sharing Account: An account dedicated to sharing stories published by other sources
- Club Journalist: A journalist that works for a specific football club
- Large Media Company: A well-known media company

Rank	Source	Type of Source	Trust Value
1	F1	Freelance	1.0
2	A1	Algorithm	0.98
3	F2	Freelance	0.93
4	S1	Story Sharing	0.93
5	S2	Story Sharing	0.93
6	C1	Club Journalist	0.93
7	M1	Large Media Company	0.92
8	M2	Large Media Company	0.92
9	F3	Freelance	0.90
10	C2	Club Journalist	0.89

Table 13 : Top 10 Sources Identified

The result does not show any clear pattern. It does not show that a particular type of source seems to be the most trustworthy. What is interesting from the result is the presence of a Twitter account that published the most trending news according to an algorithm(A1). This source was not limited to publishing only sports news. The fact that a source that simply publishes the most popular stories of the day made it to the top 10 perhaps indicates that the algorithm applied is not sufficient given that it does not punish sources for making incorrect claims once the correct claim is verified. If a source were to simply repeat all the claims made that day so far, it would be a quickfire way for them to get a good trust value. Some might also question the result based on the fact that very little large reputable media companies made the top 10, however a simple explanation for this reason is the way the algorithm was set up – It did not recognise claims made the same day that a transfer officially occurred, which would work against top news corporation who often only publishes stories when the transfer has officially occurred. The presence of a club journalist in the top 10 will have been influenced by the activities of that certain club in the transfer window, as a club making many transfer will make it is easier for that journalist to obtain a greater score.

Further analysis of sources

The benefit of the system implemented is that a great deal of analysis can be done on any source. For example, the three figures (15,16,17) below shows how sources can be compared against each other on a number of different properties. While the system at present can't format the output to display such figures, it gives all the necessary data for this to be done manually, and expanding the system to be able to do this, isn't a difficult task as everything is already in place. The charts below refer to the sources used in Table 13 while S2 and F1 are the same source from Table 12 & Table 13.

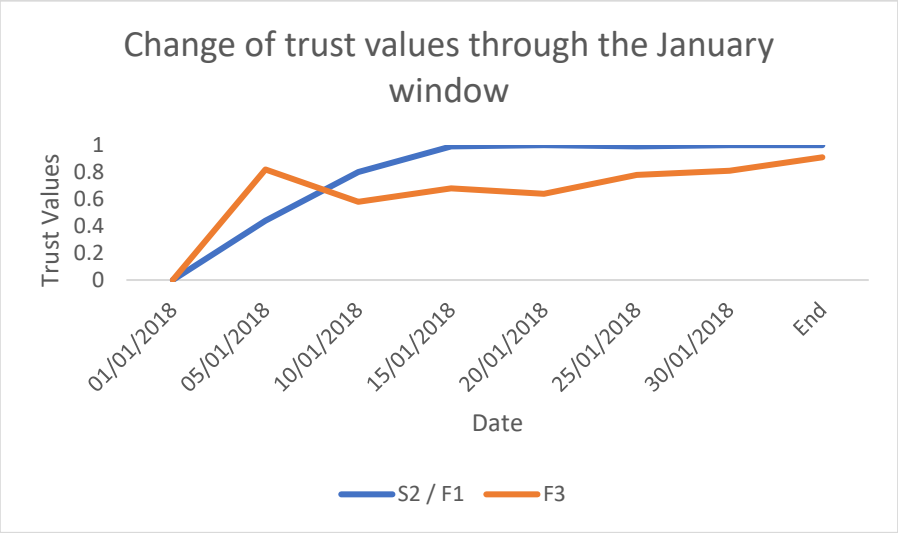

Figure 15 : Comparing 2 Sources - Trust Value

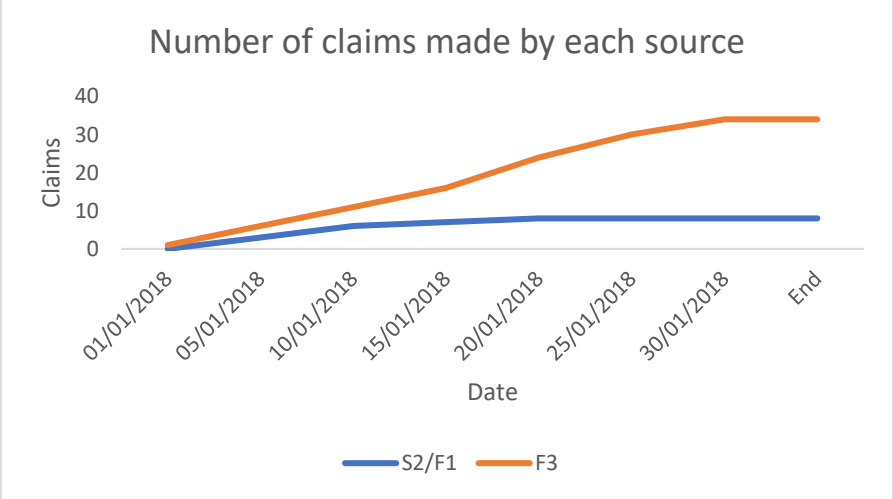

Figure 16 : Comparing 2 Sources - Claims

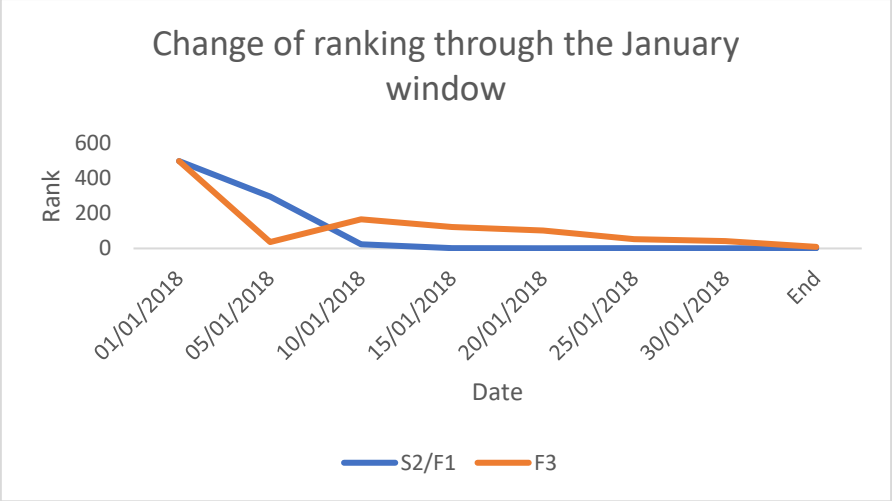

Figure 17 : Comparing 2 Sources - Ranking

The graphs show that it is possible to analyse different sources over the course of the transfer window, and see how their score and ranking fluctuates over time and how often and when the sources make their claims. Just from the three figures above, we can deduce that the Trust score of F3 fluctuates quite often compared to S2 which had a more consistent steady score. Evidently this could be because of the increase in claims made by F3 as the window went on. Given that S2 retained its score even though no new claims were made in the latter stages, it is clear that the source had made claims that became more popular as the window went on. Figure 17 above shows the ranking of the two sources as the window went on (Top 500 shown).

5.3 Top Claims

The system not only outputs the reliability of sources, it also outputs a belief value for the different claims made. This allows us to track different claims throughout the window to spot any developments. Interestingly for this specific project, we can also see which claims had high belief values, but did not follow through with a transfer before the window shut. Table 14 below shows the 10 claims with the highest belief values that did not end in a transfer.

Claim	Number of sources making claim	Belief value of claim
Riyad Mahrez - Other	1,632	0.73
Eden Hazard - Other	1,177	0.69
Emre Can - Other	1,220	0.57
Daniel Sturridge - Other	614	0.46
Harry Kane – Other	1110	0.46
Daly Blind - Other	527	0.39
Andreas Christensen - Chelsea	487	0.37
Juan Mata – Other	555	0.37
Jonny Evans - Other	709	0.36
Jonny Evans - Arsenal	527	0.35

Table 14 : Top 10 Claims

There are quite a few things to evaluate from these results. Although Riyad Mahrez – a Leicester City player did not transfer to any team in the January window, there was heavy rumours surrounding the possible transfer, and it was claimed by many sources that the player himself was trying to force a move. Therefore, a high belief for this claim isn't very surprising, similarly for most of the players listed, there were strong rumours of Eden Hazard refusing a contract to force a move away. It is a similar story for all but 1 player in this list. The surprise result was Andreas Christenses as he was already a Chelsea player. A simple reason for this error was that the player was loaned out and returned to his club in the January window, but his current team was not mentioned as Chelsea in the input files.

One issue that will need to addressed is the relationship between a team T and the team 'Other'. They are treated as completely separate values in this system, while that is not completely true. Table 14 shows Jonny Evans's potential move to 'Other' and 'Arsenal' to be completely separate, when in fact it is some form of overlapping claim. The issue will need to be addressed in any future work.

Further analysis of claims

Below is an example of the possible analysis that could be done on different claims. It compares how two different claims came to be believed over the transfer window. The claims were:

- Riyad Mahrez : Move to 'Other' i.e. move away from Leicester (Deal never happened)
- Olivier Giroud : Move to Chelsea (Deal officially happened on January 31)

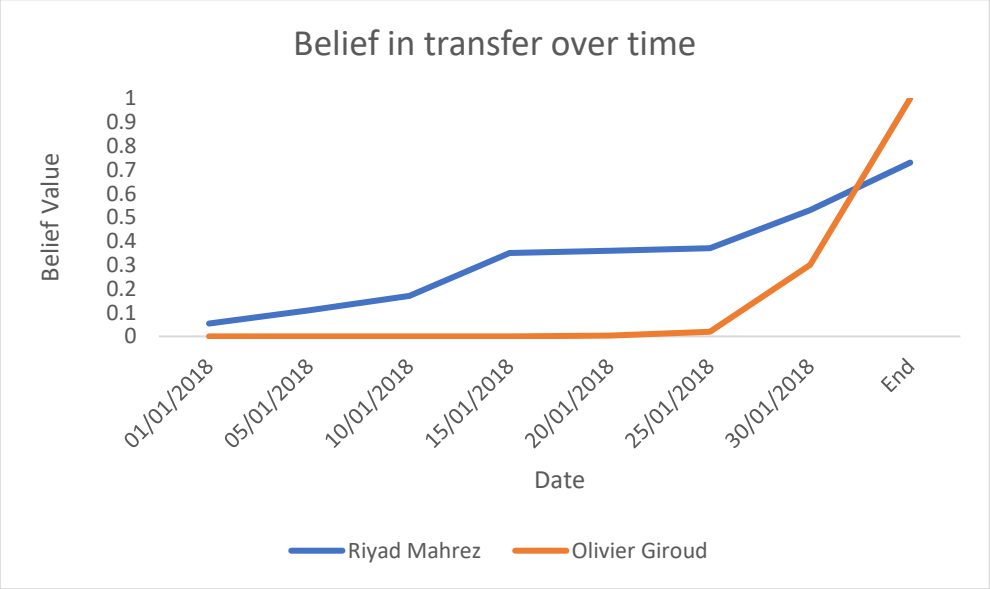

Figure 18 : Belief in transfer over time

Figure 18 above shows us how slow or how sudden a transfer can happen. In this case, there is a consistent belief that Mahrez will transfer and that belief is slowly increasing over time. Eventually the transfer did not happen. However, in the case of Giroud, the belief that he

will transfer is virtually 0 until right at the end of the window when there is a sudden spike of belief up until the last day where the transfer happened

These were just examples of what can be analysed from the huge wealth of data gathered and processed. As the system will run and process data from more transfer windows, its accuracy will improve. This section really is just a brief summary of what can be evaluated from the data made available in this project.

6 Future Work

It was regularly mentioned throughout this research that it was intended as a starting point – a proof of concept that shows the feasibility of such a system. The project was essentially split into three different parts – Data gathering, NLP & ML to extract the relevant information and an algorithmic approach to computing Trust and Belief. Neither of these three parts were implemented perfectly, there was a 26% error rate in the Naïve Bayes classification, a whole host of constraints and limitation on identifying players and teams mentioned in each tweet, and the algorithms implemented to calculate a trust and belief value were heavily simplified and did not take into account many factors when making claims. While the system has provided us with results to evaluate, to publish these results with credibility, there needs to be an improvement in the way the systems identifies claims and computes the reliability of these claims. Below is a list of the assumptions, constraints, weaknesses and areas in need of improvements. The list summarises what has already been discussed throughout the project, along with some general recommendations about moving forward with this project in the future.

Data Capture

Because of the nature of the football transfer window, the data comes in one large bulk twice a year, this posed a problem for this project as it had to process large amounts of data in a short space of time. As has previously been mentioned, the long-term aim of the project is to deal with this data as soon as it is captured in real-time. This was not feasible for this project because of the timing of the transfer window, however this should not be a problem for any future projects. The only improvement to make in the future would be to merge this step with the data extraction step and deal with filtration and classification of the data as soon as it comes in, to avoid saving large amount of unwanted data. This would improve on the efficiency of the current system implemented.

NLP and ML improvements

The NLP and ML approach taken was heavily simplified, this was mainly due to the time constraint placed on the project. The system can only extract a claim from a tweet that is directly referring to a single player. The system does not take steps to ‘understand’ what is being said, but rather look for distinguishing features so it can decide whether the tweet is about a possible move or some other news not containing a claim regarding a transfer. If the system has decided that it matches the features of a typical tweet containing a claim regarding a transfer, then it will match the player identified (IF 1 and only 1 player is mentioned) to the team identified (again IF 1 and only 1 team is mentioned). In the case where no team is identified, the team will be regarded as ‘other’, which essentially means that the claim states he will be playing for a different team to the one he currently plays for when it comes to the end of the transfer window. Again, this is an unrealistic assumption because for example a source might be tweeting about common transfer rumours regarding

a player moving away from their current team, but their tweet could actually be claiming the player will stay put and not move away from their current team – The classifier implemented might mean that this will be classified as a transfer tweet, and the methods taken to identify the player and team involved could pick this tweet up as a player P moving to team Other. For a fully credible system in the future, steps must be taken to tackle issues like these and ensure the system better 'understands' what the source is trying to claim in a tweet. Some methods to tack these issues could be:

- Improving the classifier – Using a different method to the Naïve Bayes method that produces better accuracy
- Better NLP techniques to take into account the possibility of sources mentioning more than one player or team in their tweets
- An additional classification process to categorise whether a claim means that the player **WILL** move or **WILL NOT** move.

Computing Trust and Belief

The algorithm implemented to achieve the final reliability values did indeed give trust and belief values. Because no previous work of this kind has been done, it is impossible to compare the results to truly know how credible they are. However, one can theorise about how good the results are based on the equations used. The equations used were significantly simplified, they failed to take many factors into account, and made many assumptions. Improving this system in the future will mean formulating new and improved equations that would also take into account additional factors such as:

- The number of claims a source has incorrectly claimed
    - Penalise a source for incorrectly making a claim – The penalisation factor could increase / decrease based on how popular the claim was i.e. if the claim was widely claimed by many sources this would result in less penalisation
- The timing as to when the claim was made
    - The greater the time difference between when a claim was made and when it actually occurred would mean a greater increase in Trust value
- Additional Time extraction for a claim
    - For every claim a new Time variable would alter the computation of trust and belief – A source can say that a transfer will be completed by a certain time

Efficiency

As with any Computer Science problem, efficiency is always a factor and should always be considered. The implementation of this system was not done with a great deal of consideration towards efficiency. This could be seen with the absence of efficient search algorithms when searching through player and team files. Any future work should aim to improve this issue urgently by implementing a search algorithm to replace the current method of simply iterating through the entire file.

Widening the scope

Before designing the solution to the problem, some clear constraints were imposed on the solution to shorten the scope of the problem. These were listed in chapter 2.5. Once the methods of the system are improved as previously discussed, future work could then look to expand the scope of the system by allowing it to work with more data from other leagues, not just the EFL.

7 Conclusions

The overall aim of the project was to propose an idea that hopes to tackle the mistrust within the world of football transfers. As the project went along, it was split into three different areas of implementation. A discussion on the technical requirements of each section was included in chapter 3. These were an elaboration on the aims and objectives proposed in the initial plan (These can be seen in Appendix B). The constraints and assumptions that were applied at any stage were also discussed in the design stage – again in chapter 3. After the different areas were implemented, intermediary results along with the problems that were encountered during implementation was discussed in chapter 4. Below is a summarisation of the technical achievements of this project:

- Created a script that streams Twitter data and pulls tweets containing any of the keywords that is passed to that script. The JSON formatted tweets are then saved into a .txt file. A separate script then extracts the necessary data from this JSON file into a separate .csv file.
- A Python script was implemented to extract formal statements from every tweet which identifies a player P and a team T that the user U claims he will move to. This used NLP and ML techniques to identify each variable and to classify the tweet as a relevant tweet containing transfer news.
- A further script was implemented to compute the trust and belief scores of each source and claim based on whether their claims had become true, the amount of sources also making the same claim, and whether their claim was valid according to a set of constructed rules.

All these steps helped to create a functioning system, which outputted the trust and belief of all claims and sources identified, which could be used by any Twitter user to help them evaluate transfer news and whether or not to believe any specific transfer rumour.

When comparing the achievements of the project against the initial aims and objectives set out (Appendix B), it is evident that aims 1 – 5 have been successfully achieved. However, there was no time left to achieve aim 6 – Research and propose new algorithms. I have briefly discussed possible ideas and where the current algorithms are insufficient, but unfortunately failed to propose and test new algorithms. Despite all this, I strongly believe that the project has been a success, and has proven that this approach towards tackling distrust in the media is definitely feasible.

8 Reflection on Learning

This final year project has undoubtedly been a massive challenge, I believe the project that I undertook was a significant challenge, considering the time allocated to finish it. I feel that splitting the project into three different sub-projects had been a very good move, allowing me to better plan what was required. However, I also felt at times that I could have conducted the entire project based on just one of these parts. This meant that I was unable to work as much as I would have wanted on parts of the project to ensure that I was able to implement something to fulfil all the requirements set. As a result, some areas of the projects were limited in their capabilities. In any future project I would certainly look to limit some of the requirements set and perhaps concentrate on less topics and look to put more work into specific parts. Given that I had no experience in NLP and very limited theoretical knowledge of ML, I did at times feel that I had given my-self too much of a challenge to get a good working system in the time given.

It is often repeated, that no matter how well planned a project is, it will always take more time than one predicts. This is something I definitely learned during this project. I repeatedly mentioned in the project that the second section (NLP & ML Classification) was going to take up most of my time, but despite this I still felt that I had spent too much time to implement a satisfactory algorithm to deal with this problem, which eventually meant that the final part of the project had to be rushed. Unfortunately, I had no time to put any meaningful thought about implementing a more specialised algorithm from scratch that would suit the system implemented better than those that were implemented in this case. It is only briefly discussed in chapter 6.

Despite this, I feel satisfied with the final system, and the result it produced. It is satisfying to have a physical output at the end of the project – A dataset of belief and trust values of thousands of sources and claims. This dataset could be analysed for hours on end. I also felt that I have learned many new skills. These range from technical skills such as NLP, along with project management skills such as better time management.

At the end of this period, I am grateful to have had the opportunity to conduct such a project and I am confident that it has greatly enhanced my abilities and my passion in this topic.

9 Table of Abbreviations

Abbreviations	Explained
API	Application Programme Interface
NLTK	Natural Language Toolkit
JSON	JavaScript Object Notation
ML	Machine Learning
NLP	Natural Language Processing
POS	Part-of-Speech
EPL	English Premier League
EFL	English Football League
CSV	Comma-Separated Values
NB	Naïve Bayes

10 Glossary

Glossary	Explained
Tweepy	A Python library to access the Twitter API
Token	A token is a string of contiguous characters between two spaces, or between a space and punctuation marks that is to be processed as a unit
Lexical Resource	A database consisting of one or several dictionaries
Lemmatization	The process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.
Stemming	The process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form.
Part-of-Speech	A category to which a word is assigned in accordance with its syntactic functions.
Stopwords	Refers to the most common words in a language that does not add to the context of a text i.e. and, it etc.
Variance	An error from sensitivity to fluctuations in the training set. High variance causes the algorithm to model the noise in the training data, rather than the intended outputs
Bias	An error from erroneous assumptions in the algorithm. High bias causes an algorithm to miss the relevant relations between features and target outputs

11 Appendices

11.1 Appendix A – POS Tagger Abbreviations

- CC coordinating conjunction
- CD cardinal digit
- DT determiner
- EX existential there (like: "there is" ... think of it like "there exists")
- FW foreign word
- IN preposition/subordinating conjunction
- JJ adjective 'big'
- JJR adjective, comparative 'bigger'
- JJS adjective, superlative 'biggest'
- LS list marker 1)
- MD modal could, will
- NN noun, singular 'desk'
- NNS noun plural 'desks'
- NNP proper noun, singular 'Harrison'
- NNPS proper noun, plural 'Americans'
- PDT predeterminer 'all the kids'
- POS possessive ending parent's
- PRP personal pronoun I, he, she
- PRP$ possessive pronoun my, his, hers
- RB adverb very, silently,
- RBR adverb, comparative better
- RBS adverb, superlative best
- RP particle give up
- TO to go 'to' the store.
- UH interjection errrrrrrrm
- VB verb, base form take
- VBD verb, past tense took
- VBG verb, gerund/present participle taking
- VBN verb, past participle taken
- VBP verb, sing. present, non-3d take
- VBZ verb, 3rd person sing. present takes
- WDT wh-determiner which
- WP wh-pronoun who, what
- WP$ possessive wh-pronoun whose
- WRB wh-abverb where, when

11.2 Appendix B – Formal Requirements initially set out

Aim 1

Process tweet dataset of JSON Documents into individual records containing date/time, username, user_ID, tweet_text. Everything else from the JSON document is to be filtered out. These records should then be further processed into a final .csv file. This file output should contain records sorted in date/time order where the oldest tweets appear first.

Objectives

- Study API Document of Tweepy Module
- Become familiar with the JSON Document format
- Create Python script that takes original dataset as input
- Implement algorithm to filter out attributes not required in every JSON Document
- Output every JSON file as a record containing the attributes : User_ID,User_Name,Tweet_Text
- Generate a CSV file, with each row containing a record generated from the previous objective

Aim 2

Using Natural Language Processing Techniques and the NLTK module in Python, extract relevant tweets from the dataset to create formal logic statements of the form : "User **U** claims player **P** will join team **T**"

Objectives

- Study general Natural Language Processing Methods
- Study API Document of NLTK Module
- Read the book Natural Language Processing with Python (Steven Bird, 2009)
- Create Python script that takes CSV of processed tweets as input
- Implement Natural Language Processing Methods using the NLTK module on python to process each twitter text attribute of every record to break down text into 2 parts (Player **P**, Team **T**)
- Output statements in suitable format (Format to be decided via further research)

Aim 3

Research suitable existing algorithms to implement a method which, at any given time computes and outputs a **Reliability** value for every twitter user in the dataset based on their past predictive statement(s)

Objectives

- Background research on similar projects
- Study methods proposed in the paper on “Knowing What To Believe” (Pasternack & Roth, 2010)
- Create Command Line Python program which utilizes the script implemented for Aim #2 and takes the output generated from that script as the input for a method to compute a reliability score
- Modify an existing algorithm to work with the input data and return a reliability value as a decimal (Between 0 and 1) for every user_ID present in the input data

Aim 4

Research suitable existing algorithms to implement a method which, at any given time computes and outputs a **belief** value for any predictive statement of the form “Player **P** will join team **T**” based on the **Reliability** values of Users **U** who have/haven’t made a statement on the Player **P** joining team **T**

Objectives

- Background research on similar projects
- Modify an existing algorithm to work with the input data and return a Belief value as a decimal (Between 0 and 1) for the selected statement (Of the form Player **P** will join Team **T**)

Aim 5

Create Python Command Line program which uses existing algorithms and from these algorithms is able to output a reliability value for every twitter user in the list of statements (.csv file) along with a belief value for any given statement of the form Player **P** will join Team **T** given that the player is registered in the English Premier League & the team is a competing team in the English Premier League

Objectives

- Implement both algorithms developed for Aims 4 and 5 in Python
- Implement a command line user interface with the following options
    - Sort Twitter Users by reliability
    - Search for Twitter User and their Reliability & History of statements
    - Calculate what was the belief value of a Player **P** moving to Team **T** at a given date & time

Aim 6

Research and propose a new original algorithm/method to determine potentially more accurate reliability and belief values

Objectives

- Research different possibilities and different methods of computing reliability and belief values
- Come up with different formulas for computing reliability and belief values and test these different formulas to obtain experimental results
- Compare experimental results with original result outputted using existing methods
- Evaluate which algorithms perform better under the circumstances

12 References

Berger, A. L., Pietra, V. J. D. & Pietra, S. A. D., 1998. A Maximum Entropy Approach. *Style-based text categorization: What newspaper,* pp. 1 - 4.

Bird, S., Klein, E. & Loper, E., 2009. *Natural Language Processing with Python.* 1st ed. s.l.:O'Reilly Media.

Carnew, S., 2017. *Social Media is a Weapon,* s.l.: USNI.

Deloitte, 2015. *Premier League transfer spending reaches a new record of £870m.* [Online] Available at: https://www2.deloitte.com/uk/en/pages/press-releases/articles/premier-league-transfer-spending-870m.html

E.T.Jaynes, 1957. Information Theory and Statistical Mechanics. *The Physical Review,* 106(4), pp. 620-630.

ECMA International, 2017. *The JSON Data Interchange Syntax.* 2nd ed. s.l.:s.n.

Edelman, 2017. *TRUST BAROMETER™- Global Results,* s.l.: s.n.

Goyvaerts & Jan, 2016. *Regular Expressions.* [Online]
Available at: https://www.regular-expressions.info/
[Accessed 2018].

Hirschberg, J. & Manning, C. D., 2016. Advances in natural. Volume Science 349, pp. 261 - 266.

J.R.Quinlan, 1986. Induction of Decision Trees. *Machine Learning 1,* pp. 81 - 106.

JLloyd, 2016. *The transfer window explained,* s.l.: Dream Team.

Jones, M. T., 2017. *Speaking out loud,* s.l.: IBM.

Katz & Doron, 2015. *Top 10 Social APIs,* s.l.: s.n.

Kleinberg, J. M., 1999. Authoritative sources in a hyperlinked. *Journal of the ACM,* 5(46), pp. 604 - 632.

Langseth, H. & Portinale, L., 2007. Bayesian networks in reliability.

M. Trivedi, S. S. N. S. S. N., 2015. Comparison of Text Classification Algorithms. *International Journal of Engineering Research & Technology,* 4(2), pp. 334 - 336.

Maglogiannis, I. & Karpouzis, K., 2007. *Emerging Artificial Intelligence Applications in Computer Engineering.* s.l.:IOS Press.

mostpopularsports, 2016. [Online]
Available at: https://mostpopularsports.net/

Pasternack, J. & Roth, D., 2010. Knowing What to Believe(when you already know something). *Proceedings of the 23rd International Conference on Computational Linguistics,* pp. 877 - 885.

Research Now, 2016. [Online]
Available at: https://www.researchnow.com/

Roesslein & Joshua, n.d. *Tweepy Documentation.* [Online]
Available at: http://tweepy.readthedocs.io/en/v3.5.0/
[Accessed 2018].

Rouse, M., 2017. *Smart Speakers,* s.l.: WhatIs.

Scoial Media Marketing LTD, 2017. [Online]
Available at: https://social-media.co.uk/

Sebastiani, F., 2002. Machine Learning in Automated Text Categorization. *ACM Computing Surveys,* Vol 34(1), pp. 1 - 47.

Sistilli, A., 2017. *Twitter Data Mining: A Guide to Big Data Analytics Using Python.* s.l.:s.n.

Smith, R., 2017. *The Original Fake News: Soccer Transfers,* s.l.: The New York Times.

Statista, 2017. [Online]
Available at: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

Stigler, S. M., 1983. Who Discovered Bayes' Theorem. *The American Statistician,* 37(4), pp. 290 - 296.

The Premier League, 2017. *Transfers.* [Online]
Available at: https://www.premierleague.com/

Timothy Baldwin, e. a., 2013. How Noisy Social Media Text, How Diffrnt Social Media Sources?.

Twitter Inc., 2017. *Developer Agreement and Policy,* San Fransisco: s.n.

Twitter Inc., 2017. *Twitter Privacy Policy,* San Fransisco: s.n.

Twitter, 2018. *Docs.* [Online]
Available at: https://developer.twitter.com/en/docs/tweets/

Yin, Xiaoxin, Yu, P. S. & Han, J., 2008. Truth Discovery with Multiple Conflicting Information Providers. *IEEE Transactions on Knowledge and Data Engineering,* 6(20), pp. 796 - 808.