UPPSALA
UNIVERSITET

Developing an artificial intelligence bot that talks about football

Olof Löfving, Ludvig Nilsson and Anton Norberg

Project in Computational Science: Report

January 2019

PROJECT REPORT

Institutionen för informationsteknologi

Abstract

We have developed an artificial intelligence(AI) bot that talks about football players. The bot was made for the company Twelve Football, that has developed an algorithm for measuring performance in football. Twelve Football provides data, that acts as a base for everything the bot says. The bot can tweet and answer questions from users, using Twitter's messenger function and does not solely mention results and statistics, but can describe players and communicate interesting opinions as well. To find out what is interesting we fitted probability distributions to all different events that occur in football, in order to investigate how much a player's performance deviates from what is normal. To parse incoming questions into structured data we used Wit.ai, that utilizes natural language processing, a sub field of AI. It takes roughly two minutes for the bot to receive messages from Twitter. The total time to reply is about three minutes, which is a long time for a chat bot.

Contents

1 Introduction

1.1 Bot

A bot, formally known as a software agent, is a computer software that is designed to run automated tasks [1]. In recent times, bots have increased in popularity by the introduction of conversational bots, such as Amazon's Alexa and Google Assistant. Conversational bots are computer programs that are powered through artificial intelligence and are used on several types of platforms. These platforms includes websites, applications for mobile phones and different messenger platforms [2].

The two conversational bots mentioned above are general and work as virtual assistants, that can answer questions and perform certain tasks, by reacting to human speech. A different type of bot is a chat bot. A chat bot can potentially perform the same tasks as mentioned above but is generally more specific in its design. As an example, Whole foods has created a chat bot where one can ask about recipes and get inspiration on what to cook for dinner [3]. Duolingo has a conversational bot with which people can practice basic conversational skills, in order to learn a new language [4].

In the field of football there exist a few chat bots. *Toni* is a football chat bot that can answer simple questions such as when games are played, what the scores were or provide links with statistics about a player. *Jeff* is a different football bot developed by Sky Sport that can deliver player statistics and general news about football. However, these bots solely focus on providing already known information and do not do any analysis themselves. [5] A company that has started to develop a different type of football bot is Twelve Football.

1.2 Twelve and its product

Twelve Football is a start-up company based in Stockholm that has developed an algorithm, measuring performance in football. Their business idea is to build platforms where people can interact with this algorithm, in an interesting and fun way. Hence, Twelve has developed a website and a mobile application, containing visualization tools. With these tools, users can compare players, visualize player's performance on the pitch and investigate in what way a player has helped his team.

Twelve's algorithm is based on assigning points to every individual event in a game of football. By summing up these points and associating them with a player, the performance can be evaluated over different time spans and in different categories, such as *defence* and *attack*. The algorithm is further described in Section 2.3.

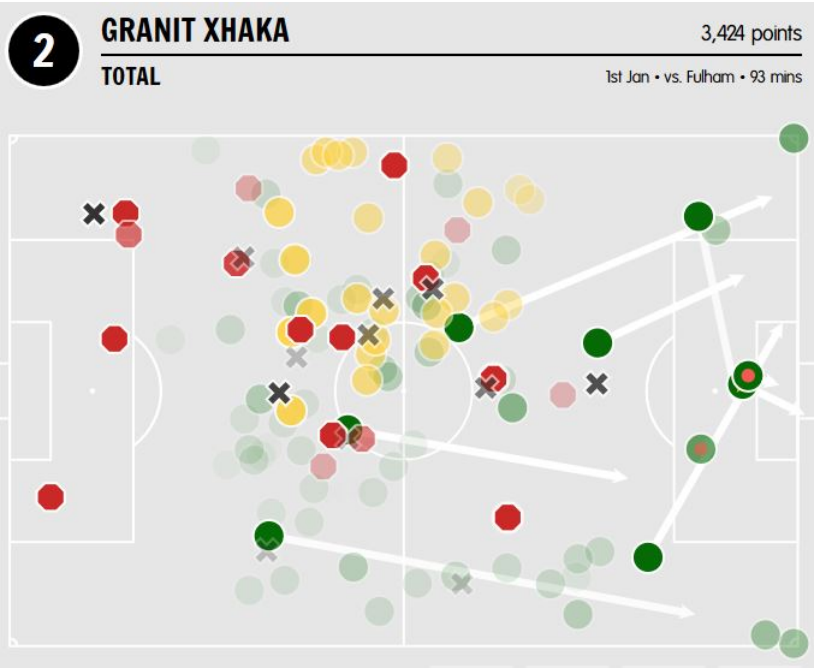

Figure 1: Visualization from the Twelve website. Each dot represents an event that a player has been involved in, during a game of football. The color represents what weight, the event is assigned to. The transparency of the dot correlates to the number of points according to the algorithm. Arrows show the direction and length of particularly interesting events and the crosses represent events assigned with negative points.

Figure 1 shows an example of a visualization from the Twelve website. By clicking each event in the picture a user will see what time the event occurred, how many points it was assigned, and what type of event it was.

1.3 Bot Development

As an additional tool to the app and the website, Twelve is developing a chat bot. The plan is to make the chat bot available on platforms such as Twitter, Facebook Messenger and on Twelve's own platforms.

The main purpose of the bot is to be a fun and interesting tool to use for people. In contrast to the football bots *Toni* and Sky Sport's bot *Jeff* Twelve's bot will form opinions and analyze players, thus being far more sophisticated than its competitors on the market. These analyses, together with other information that can be retrieved from the bot, will increase the information that can be gained from Twelve's platforms and draw people into using Twelve's website and app. For pure marketing purposes, the bot will also create sentences that it posts on Twitter, together with a link to an interesting visualization on the website.

1.4 Project Goal

In this project we have implemented further developments of Twelve's chat bot. At the start of the project the bot was able to create sentences about the best player in a game, based on the player's offensive or defensive events. It could post these sentences on Twitter but it could not answer questions from users. Our two main goal has been to make the bot:

- produce a large variety of interesting and correct tweets about a player,
- answer written questions correctly.

1.5 Disposition

In this report we first cover Twelve's algorithm, probability distributions and the natural language interface *Wit.ai*. Probability distributions are used in order to analyze players and *Wit.ai* is used to parse incoming questions into structured data. Thereafter, the method of development is explained, in which we distinguish between Twelve's developing process of the bot and what has been done in this project. We then show how the bot works through several examples and conclude with a discussion about issues and possible future improvements.

2 Theoretical tools and framework

In this section we discuss background theory of the tools and frameworks that we have used. This includes explaining the process of evaluating players by using Twelve's performance algorithm and how probability distributions relate to the game of football. First, we explain how a computer program can interpret and understand human language by using *Wit.ai*.

2.1 Wit.ai

Wit.ai(Wit) is a free software used to parse natural language, i.e. human written text and speech, into structured data. This process is called natural language processing (NLP) and is considered a sub-field of artificial intelligence. The specific model of NLP that Wit has developed is based on so-called entities, see [6].

The most important information in an entity can be divided into a value and a type. Wit has defined three types of entities; *trait, free text* and *keyword*. Trait entities are used to understand the intent of a message by assessing the whole stream of natural language in the message. The value of the trait entity belongs to a predefined list defined by the developer that uses Wit. Keyword entities are specific words in the message, whose value belongs to a predefined list in Wit, also decided by the developer. Free text

entities are extracted using a sub-string of the sentence that are not in the predefined keyword list and doesn't define a purpose. Free text entities can take any value and are found by using look up strategies, see [6].

Apart from the type and the value, each entity contains a confidence level, describing the probability, that the entity was extracted correctly. To increase the confidence level Wit can be trained, using sentences where the correct entity is specified. Once trained 50 times, Wit returns the precision and the recall over a confidence level ranging from 0 to 1. *Precision* describes how often Wit is correct when it detects the specific entity and *recall* describes how often Wit detects an entity when it is in the message. Ideally both high precision and recall are desirable but there is usually a trade off that each developer has to do. If it is important that entities aren't classified incorrectly, a high precision is desirable. If extracting all entities from the message is of higher importance than classifying them correct, a high recall is desirable, see [6].

2.2 Probability Distributions

A probability distribution describes the probability of the outcome of an event, in terms of an underlying sample space. The sample space is either discrete or continuous and a game of football consists of a number of countable discrete events. Some examples are the count of how many passes, goals and tackles that a player has done. However, certain attributes that are connected to these events are not discrete. A pass has a length which is recorded in meters and is thus continuous. Furthermore, the relation between different events take on continuous values, such as the relation between the number of backward and forward passes. Therefore both continuous and discrete distributions are related to football.

The discrete values in football are the count of events and cannot be negative. This also holds true for the continuous values. For example, the pass length will never be less than 0. Distributions that only includes values on the positive side of the real axis are thus to be considered.

When modelling the probability of a discrete event in football, such as a player scoring 3 goals in a game, a Poisson distribution is suitable [7]. For a Poisson distribution the events should be independent, which holds true in football. For example, if a player has scored a goal it does not increase his chances to score another goal. However, to model the probability of a continuous variable, such as the pass length, a Poisson distribution cannot be used since it requires the variable to be discrete. A Normal distribution can be used instead, considering the central limit theorem which states that the sum of independent random variables, tends towards a normal distribution [8]. Furthermore, when the average of a discrete event is above 15 a Normal distribution can be considered instead of a Poisson distribution [9].

When assuming that the probability of a certain value follows a probabil-

ity distribution, it's possible to calculate the probability of the value being over a certain threshold. That is, instead of calculating the probability of a player scoring 3 goals, the probability of scoring 3 or more goals can be calculated. This is done using a cumulative distribution function.

2.3 Algorithm to evaluate players

Twelve has developed an algorithm to evaluate player performance in a game. The algorithm assigns points to all events that occurs in a game, and divides them into the weights *Attack, Defence, Shot, Error* and *Press*. Furthermore, the events are assigned to specific players. In order to get an individual score for a player, all events of the player are summed up. The maximum number of points for a single event is 1000, and is awarded when scoring a goal. All other events are evaluated relative to a goal. For attacking events, such as a pass, the points are evaluated in relation to how much the action increased the chance of scoring a goal. If a pass increases the chance of a team to score by 10% percentage points, that single pass is awarded 10% of what a goal is worth, i.e. 100 points. This works similarly for defensive events. If a player recovers the ball in a position where a goal is scored 20% of the time, the player receives 20% out of 1000, i.e. 200 points. Defensive events that happen of the ball are called press events. Press events do not stop the opponent completely and therefore the points assigned to those are not as high as for a defensive event. Instead, the points are assigned as a ratio of the corresponding defensive event, based on how active the player was adjacent to the ball. Shots are evaluated on how likely the shot is being a goal. If the shot has a 50% chance of resulting in a goal, the shot receives 500 points. The algorithm also deals with the errors a player makes. The errors are within any of the other weights. It could for example be a missed shot or a failed tackle. Error points are always negative.

All actions are based on how likely the action is to result in a goal or how much the action increases the probability to score. These metrics are calculated using an expected goals model. However, we are not allowed to discuss Twelve's specific model in this report, in any further detail.

3 Implementation and Design of the bot

The project is divided into two parts, in which we have tried to reach the two goals stated in the introduction. The first part is called *Tweet* and the second *Chat Bot*. The Tweet part includes the development we did regarding what the bot can say, how to build up sentences and what underlying statistics from a football game it bases its sentences on. In the chat bot part we developed the bot's ability to answer questions, namely, interpret incoming messages and choose a replying sentence. In order to understand what has been done in the project, we describe how the bot was structured in the beginning of the project, followed by a section with the developments and improvements made during this project.

3.1 Tweet

The idea behind the Tweets is that the bot should draw attention to the Twelve website by saying something interesting about a player after each game. In order to do this the bot has to retrieve data from a football game, analyze it and create sentences from the analysis.

3.1.1 Structure of the bot at the beginning of the project

The bot is developed in Python and is connecting with the Twelve website in order to get match data to analyze. Each event in the match data is assigned points according to the Twelve algorithm and is connected to a specific player. Furthermore, each event is assigned to one of the weights *attack, defence, shot, error* and *press*. The data processed through Twelve's website comes from a company called Opta Sports, who are collecting the data manually.

By using the events, the bot builds short sentences based on facts and entities, that we call clauses. It is in the creation of the facts and entities where the analysis of the player is done. A fact is based on the count of different events and an entity is generally based on relations between different events. An example of a fact is the number of assists that a player has done in game and an example of an entity is the type of event that was most frequent in a game.

Each fact and entity is connected to a JSON structured file that holds building blocks of words that can be combined into a variation of clauses. These building blocks can be divided into *pre-clauses* and *arguments*. For example, from the fact that a player has done two assists, the pre-clause is building blocks such as *contributed with, made* and *provided*, while the argument will be *2 assists*. In this example the fact clause about assists can be *provided 2 assists.* Entity clauses are built in the same way. Taking the example with the most frequent event being a pass, the bot can produce

an entity clause being *helped in the attack by making a lot of passes*, where *helped in the attack* is the pre-clause and *making a lot of passes* is the argument. To build complete sentences the clauses are combined, using context clauses, a player name and fill words such as *and*. These combinations can be done in a variety of ways and one possible combination can be seen in pseudo equation (1).

$$\begin{aligned} Sentence = {} & ContextClause + PlayerName \\ & + FactClause + FillWord + EntityClause \end{aligned} \tag{1}$$

Following equation (1), the fact clause and the entity clause are combined using the fill word *and*. Then put into a context using a context clause and the name of the player the sentence is about. An example of context clauses are *in the match against Manchester City* an *over the last season*. Using the Liverpool player Mohammed Salah in the example, the final sentence may end up as; *In the match against Manchester City, Mohammed Salah provided 2 assists and helped in the attack by making a lot of passes.*

The bot always produces every possible entity and fact clause in each category of events, and puts them into a list. Thereafter the clauses are chosen from that list and put together by following a scheme such as shown in equation (1). At the start of the project the choice was random, meaning it did not matter if a player had done zero assists in a game. The bot could potentially create a sentence about it anyway. Furthermore, the bot was programmed to tweet about the player with the highest total amount of points, or the highest points within the weights *attack* and *defence*. The bot was scheduled to tweet right after a game was played, in the English Premier League.

3.1.2 Development of the tweet function

We developed the tweet function of the bot with the goal to produce interesting and correct tweets. To achieve this, we started to extend what weights the bot could talk about, to include *Errors* and *Shots* as well. This work included extending the counter for specific facts and implement entities for the new weights. Furthermore, we created new JSON files that contained building blocks in order to make new clauses. The weight *Error* contains events that are assigned a negative value, since it is not helpful to commit an error. In order to make it interesting to tweet about an error we extended which player the bot could tweet about, to include the player with the most amount of negative points. Furthermore, we removed the case when the bot tweets about the player with the highest amount of total points. This was done because that player usually is the one that has scored, and in the long run it is not interesting to only talk about goal scorers.

To produce interesting tweets, we decided to evaluate players compared to what is normal. To achieve this, we created distributions, for the count of all different types of events that a player can be involved in, over one game. The data comes from a season and a half of Premier League and Champions League, which results in 750 games containing over 1.5 million events. Players that have played few minutes would have few events and thus not be good representatives of what is normal in a game. Therefore, a threshold of 60 minutes play time was added.

In a game of football, players have different positions, that focus on different things. It is usually the forwards that score the most and the defenders that contribute with the most clearances. This means that what is considered normal depends on the player's position on the pitch. Therefore, we created distributions focusing on the four major roles; *goalkeeper, defender, midfielder* and *forward*.

Once the events were counted and divided into roles they were fit to probability distributions. One could have tried to fit probability distributions to each individual data set, but due to the vast amount of different categories of events in a game a Poisson distribution was chosen to model the discrete events and a normal distribution to model the continuous events. An exception was made when the mean of a discrete event was larger than 15. It was then approximated as a Normal distribution, according to the theory in Section 2.2. Using a normal distribution can also be motivated by the discussion of the Central Limit theorem held in Section 2.2. The decision to only fit to two probability distributions, greatly simplified the coding process.

One could argue that it is not enough to consider only two probability distributions. However, we believe that the advantages of simplifying the coding process and keeping it general is greater than the disadvantages of having some data sets badly fitted.

The next step was to use the distributions to make the bot choose facts and entities that are interesting to talk about. In order to do this we defined *interesting* as something that deviates from what is normal. As an example, scoring four goals in a game is considered interesting since it happens rarely. If that happens, it should be likely that the bot tweets about it. When choosing which fact to tweet about the value of the player in a specific game is compared with the average value for the position of the player, creating a weight for that specific fact. This way, facts that are high, compared to the average value, are more likely to be chosen.

We developed the choice of entity in a different way, using the distributions of the events and the values that the different entities are based on. The underlying value of an entity was divided into the categories *low, medium, high* and *exceptional* and was given a weight according to what category it belonged to. *Low* was assigned if the value was below the 17th percentile of the distribution, *medium* between 17th-83rd, *high* between 83rd-95th and

exceptional above the 95th percentile. To calculate the percentile boundaries when the data set was assumed to be Poisson distributed, the cumulative distribution function was used. When the data set was assumed to be normal distributed, how many standard deviations away from the mean was used instead. This is essentially what the cumulative distribution function calculates.

When using this system to categorize the values and if there are more than one game that the entity should cover, the boundary values have to be modified for the normal distribution. This is done by dividing the deviation of the boundary from the average with the square root of the number of games. The 83rd percentile represents one standard deviation from the mean in a normal distribution. Therefor the boundary for *high* can be written as:

$$Boundary_{high} = average + \frac{std}{\sqrt{games}} \tag{2}$$

where games is the number of games the entities are based on.

Generally the different categories were given the weights 0.5, 0, 1 and 2 in increasing order from *low* to *exceptional*. The higher the value of the weight, the more likely it was that the bot included it into a sentence. Furthermore, we gave the entities a Boolean value, telling if it was significant or not, to tweet about. Generally, only entities that were high and exceptional were assigned to be significant.

To make sure the bot tweeted correct things we had to look carefully into the wording of the produced sentences. For facts it was straightforward since there is nothing contradictory in telling about the number of events that are connected to a player. However, when writing building blocks for the entity clauses we had to be more careful. Taking the example with the most frequent type of event that was discussed earlier, the bot produced the clause *helped in the attack by providing lots of passes*. That sentence may seem to make sense but we cannot be sure it is correct. Passes are the most common event in football and to assume that a player has produced a lot of passes, just by comparing to other events, may not be correct. Therefore we modified all entities and the building blocks in the JSON files, so that we were sure the clauses produced, were correct.

In order to solve the problem above, we changed the underlying value in the entity. Instead of being based on the count of each type of event, we based it on the number of points generated by that type. Furthermore, we changed the building blocks so that the bot produced clauses of the type *got most ranking points from making passes* or *mainly helped his team offensively by making passes*. To still be able to say that a player has done a lot of passes we created a new entity only based on the number of passes and with the categories, *low, medium, high* and *exceptional*. If the count of passes are within the category high or exceptional we can say that it is a lot of passes, since it is compared with the distribution of passes in general.

3.2 Chat bot

The chat bot part of the project involves developing the bot to answer incoming questions, specified by users.

3.2.1 Structure of the bot at the beginning of the project

The major parts of the chat bot had been outlined in the beginning of the project. However, a number of parts needed to answer a message had not been created.

To make the bot being able to interpret incoming messages from a user, the natural language interface wit.ai (Wit) is used. Wit takes in a message and returns a list of entities containing type, value and confidence level. The types of entities are specified by the developer using Wit and can be in the three categories *trait, keyword* and *free text*. Wit is further described in Section 2.1. The entities developed on Wit are *intent, game span, teams, weight* and *contact*. Intent and game span are trait entities. The value of the intent tells what type of question the bot should answer and the value of the game span specifies the different number of games that the question intend. Teams and weights are of type keyword entities. The team entity includes all the teams the bot can talk about and the weight entity specifies the different weights *error, press, attack, defence* and *shot*. Contact is a free word entity, which tells what player the question is about.

When Wit is returning an entity the bot responds to it, according to its type and value, with corresponding functions. At the beginning of the project a framework was in place but the functions were not fully implemented. Furthermore, Wit was not properly trained, having low precision and recall when interpreting messages into entities.

There is a database for storing player names and teams with corresponding unique ID's. The ID's are used, instead of names, when getting data from the Twelve website. At the start of the project the code for the database was working locally but it was unstable and could not run on a remote server. The database also stored all message ID's from Twitter, so that the bot knew which questions that already had been answered.

3.2.2 Development of the chat bot

The main goal with the chat bot was that is should be able to answer written questions correctly about a player. Furthermore, we wanted the bot to answer questions with a descriptive answer about the player, not just mentioning stats. We developed the bot so that it can handle questions about who has been the best or worst player, statistics about a player, comparing two players and giving an opinion about a player. These questions can be answered within one of the weights attack, defence or shot, over

different game spans including a specific game, an entire season, last game and recent games.

To be able to answer questions correctly we first had to make sure that the messages were parsed correctly on Wit, by developing its entities. The application developed on Wit included intent, game span, team name, player name and weight. However, these entities did not contain the correct values and many of them were redundant. Too many values makes it harder for Wit to classify entities correctly, thus we removed unnecessary values. We added the entity *League* to be able to distinguish between different contests within football and the entity *stats* to tell what specific fact a question intended. Stats is a keyword entity that we defined as the name of all the different facts the bot can base its sentences on. The stats entity is only used when the intent is to get statistics about a player.

We made implementations to verify that messages are parsed correctly on Wit and to deal with the cases when entities are missing. The two entities that is necessary for producing a reply is intent and game span, meaning the bot had to make assumptions if they were missing. Those assumptions are made depending on the other entities picked up by Wit. For example, if the intent entity is missing and the bot only got one player entity from Wit, the user who sent the message probably wants an opinion about that player. If game span is missing, the bot assumes the user is interested in the recent games, which we defined as the last three matches. This assumption is made, since it is interesting to know a players current form, and one game is not enough data to base that on. Furthermore, deciding the current form from the entire season is excessive and takes a longer time to calculate. If no league entity is found, the bot uses the league in which the player or team mentioned in the message belongs to.

To catch player entities, in which the value is misspelled, we implemented a spellchecker. The spelling needs to be correct in order to find the correct player in the database containing their ID's.

When the entities are gathered from Wit the bot collects match ID's depending on which teams, players and game span the question is parsed into. These match ID's are used, together with the player ID's, to get the correct match data from the Twelve website. When having the match data, the process of building a sentence follows the same principle as explained section 3.1. However there are more options of what the bot could say, which is decided by the other entities from Wit.

If the questions are not parsed into a specific player, the bot choose a player depending on the intent, weight and game span. For best and worst player this means the player with most points or the highest amount of negative points, respectively. If the intent is the worst player, we implemented that the goalkeeper is ignored, since conceding a goal gives a large amount of negative points. This means goalkeepers are highly overrepresented when choosing the worst player from negative points. The sentences produced are

the same as for the tweets and cover a specific weight. If there is no weight entity, the bot will talk about the weight in which the chosen player gained the highest amount of points. This is because the player will generally have more events in that weight and thus the bot can produce a larger variety of interesting things to talk about.

For all intents except player statistics, the sentence is generated in the same way as for the tweets. If the intent is player statistics the bot makes a sentence of the requested fact instead.

One aim with the chat bot is to draw people into the Twelve website. Therefore, we included a link to the website with a pitch preview, with the answer.

When generating a sentence or parsing a question, errors may arise. To improve the user experience, we implemented error messages, explaining what went wrong, that are sent to the user if an error occurs. For example, sometimes the bot can not find the specific player in the question, making it impossible to answer. It then replies with *Couldn't understand which player or teams you meant. Please try something else.* There are also unexpected errors that may occur. If that happens a general error message is sent to the user. The user should always get a message so it knows that the bot is running.

The bot does not have an endpoint from Twitter to tell it when there is a new message. Therefore, we made a scheduler that request messages from Twitter every minute. The reason that we only check once a minute, is due to limitations specified by Twitter. When the bot does a message request it collects all the messages sent to the bot and compare the message ID's to the database, where all answered message ID's are stored.

4 Results

In this section we cover examples of how the bot performs when tweeting and answering questions. We give two examples of tweets and four examples showing how the chat bot answers questions with the intents *best player/worst player, player statistics, player opinion* and *compare player*. An example of what happens when the bots does not understand a message is also shown. These examples does not cover everything that the bot may talk about but show the general capabilities of the bot.

4.1 Tweet function

During the project we increased the number of facts that the bot could talk about, from 9 to 25 and the number of entities from 18 to 35. Below we include two examples of tweets made by the bot.

Figure 2: Example of a tweet within the category defence.

Figure 2 shows a tweet about the defensive performance of Aron Gunnarson, against the team Tottenham Hotspur. The tweet is built up by the fact *ball recoveries* and an entity based on the number of points he collected according to the Twelve algorithm. The underlying value in the entity was categorized as *exceptional* and therefore the bot said he got amazing defensive points. Aron was chosen because he had the most defensive points in the game and the picture is a link to the Twelve website, containing a visualization of his defensive events in the game.

Figure 3: Example of a tweet within the category attack.

Figure 3 shows a tweet about the offensive performance of Andrew Robertsson in the game against Manchester City. The tweet is built up by the fact *passes* and an entity based on the average point he received for every pass. He was averaging a number of points that were categorized as exceptional. Andrew was chosen because he had the highest amount of offensive points in the game. The picture in the link to the Twelve website shows a leader board based on the offensive point received in that game.

In the first examples the bot worked well. It chose an entity within the category *exceptional* and the fact was based on a large count. This is the case in the second tweet as well but there are problems that enlighten the two major issues with the tweets. The first one is that the entity clause and the fact clause are about the same type of event, in this case *passes*. In Figure 3 it is a problem since it makes the wording of the sentence unnatural. In other cases it is even clearer. An example is the sequence *provided 50 passes and made a lot of passes*. Besides being an unnatural way of writing, the second part of the sentence is uninteresting since it essentially has the same meaning as the first part.

The second issue is that the sentence is phrased in a way that we cannot be sure is correct. The phrase *several exceptional passes* indicate that the player has done a lot of passes, while the entity builds on the average value

of the passes made. In this case it is true, but if the player only made one pass in a game it would not be correct.

4.2 Chat Bot Function

The chat bot has been developed so that it can categorize the sentences into the intents *best player, worst player, player statistics, player opinion, league* and *compare player*. It can answer questions about players from Premier League and Champions League based on the current season.

Figure 4: Question asked to the chat bot on Twitter. The minute notations states when the question was asked and when the answer was delivered.

As can be seen in Figure 4 the question was answered correctly and with a link to the Twelve website. Looking at the timings we can see that the bot took approximately 3 minutes to answer the question. Furthermore, we see that the whole name *Mohammed Salah* was not necessary for the bot to pick up which player the user meant. The bot picked up the intent of the question as *player statistics*.

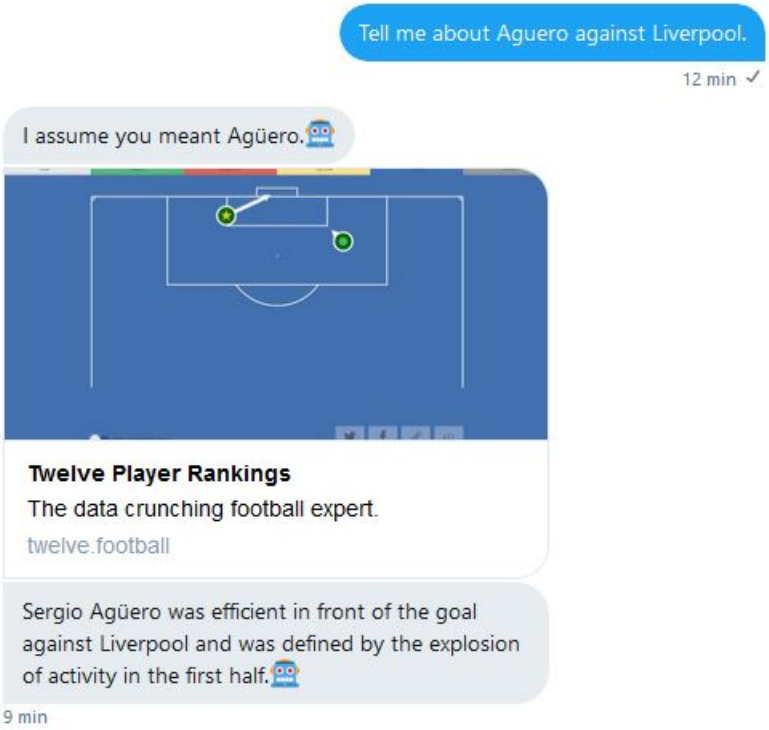

Figure 5: Question asked to the chat bot on Twitter. The minute notations states when the question was asked and when the answer was delivered.

As can be seen in Figure 5 the chat bot notices that the name is not spelled correctly and notifies the user, which player it thinks he or she meant. Since the question is not about something specific, the bot interprets the intent as *player opinion* and answers in the same way as when it tweets about a player. The answer is built on two different entities. The first one is based on a comparison of the accumulated probability of scoring and the number of goals a player has made. In this case the goals exceeded the probability of scoring with over 0.5 number of goals and therefore the bot said that Agüero was efficient in front of the goal. The part about *the explosion of activity in the first half* is based on an entity that compares points gained, between different time periods.

Figure 6: Question asked to the chat bot on Twitter. The minute notations states when the question was asked and when the answer was delivered.

In Figure 6 we see a question with the intent *worst player*. In the message, league in Premier League is spelled incorrectly as leuge and there is a space in the word is, however the bot is still able to interpret the message correctly. Sean Morrisson was chosen as the worst player in the Premier League. A foul that leads to a penalty is the mistake that gives the highest amount of negative points and according to the bot he has done a lot of things wrong. That iss based upon counting the number of events assigned with negative points. The link to the website does not show a picture of a visualization. This means that something went wrong when taking the screenshot that should have been in the link.

Figure 7: Question asked to the chat bot on Twitter. The minute notations states when the question was asked and when the answer was delivered.

In Figure 7 we see a question where the intent is to compare two different players. The bot chose Ronaldo since he got the most points over the last three games. The explanation why Ronaldo is the best is based on the same principles as when the bot tweets about him. That is, the bot chooses the category where Ronaldo has the most points and combines fact and entity clauses. Since there is not a time specified by the user, the bot looks at the three different games played by Ronaldos team.

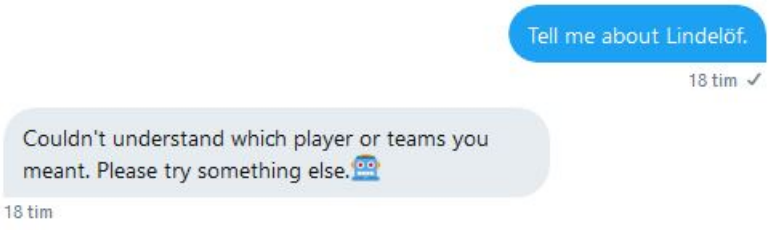

Figure 8: Question asked to the chat bot on Twitter. The hour notations states when the question was asked and when the answer was delivered.

As can be seen in Figure 8 the bot is unable to respond when asking about the Swedish player Lindelöf. This is because of the Swedish letter *ö*, which makes it hard for the bot to understand which player the user meant.

5 Discussion

In the discussion section we highlight issues with the bot and discuss how the bot could be changed to fix them. Furthermore, we bring up suggestions on how to further develop the parts of the bot that work well.

5.1 Tweet

In Section 4.1, we stated two issues with the bot. The first one was that the sentence produced by the bot may contradict to what actually happened in the game. This issue occurs when basing the sentence upon an entity, and the wrong conclusions has been drawn from the underlying fact. The example we gave was that the bot said a player *provided several exceptional passes* while the underlying value was the average points gained for each pass, and not the number of passes. This issue rarely occurs when the bot tweets, since it bases the tweet on the player with the highest amount of attacking points. That player has almost always provided a lot of passes. However when the entity is used by the chat bot the issue will occur more often, since the user might ask about players with few passes made.

One way of dealing with this issue is by rewriting the building blocks in the file connected to the entity, so that it is not possible to make that error. This is also the way we have worked during the project. Another way is to change the underlying fact in the entity so that it better fits with what the bot is saying. In the example mentioned, one could change the significance to *false* if the number of passes is below a certain threshold.

These issues are hard to detect as the development of the bot goes on, since the difference is small between what is correct and what is not. To keep these errors at a minimum, one has to be thorough when creating new entities, and when changing them. Furthermore, the sentences that the bot creates needs to be evaluated in depth by manually comparing what the bot has tweeted, and answered, with the actual events in the game.

The second issue is that different parts of the tweet is about the same thing. One example is the sequence *had several exceptional passes and made a lot of passes*. One solution is to give a label to all the facts and entities. If the labels are the same, for a fact and an entity in a sentence, it is discarded by the bot and replaced with another sentence.

An improvement that could be considered when creating sentences is incorporating the category low when choosing an entity to include in the sentence. This could be interesting when talking about the worst player. As of now, the things the bot can create sentences about, when talking about the worst player is fairly limited. This is due to the fact that there has been a focus on players that preform well. It would thus be interesting to incorporate low to increase the number of sentences and also describe that a player was the worst because he did not contribute with important actions

for his team. It could also be interesting to incorporate low when creating opinion based sentences about players. This would work as a complement to the things a player did well and serve as something the player could improve on, ultimately leading to a more realistic and interesting answer.

5.2 Chat bot

As stated in Section 4.2, it takes about 3 minutes for the bot to answer a message on Twitter. Most of the reply time comes from two sources. Firstly, it takes up to 2 minutes for the bot to register new messages. Secondly, the bot can only request new messages from Twitter once every minute. The actual run time for the bot is 20-30 seconds, which is quick compared to the time it takes to get messages from Twitter. However, the run time increases when a reply involves a lot of games to process. This poses a problem for a chat bot because it might be hard to attract regular users, when the waiting time is too long.

One solution to the problem is to request the messages from twitter with a higher frequency. However, that costs money and would require an investment decision from Twelve Football. Another way is to launch the bot on another platform, by changing the way it is receiving and sending messages. In that case there could be a direct endpoint between the platform and the bot, making the bot able to start an instance as soon as the message is received. Furthermore, the instances can be run in parallel, meaning the bot can answer several questions at the same time.

To reduce the run time of the bot, it could download data for the whole season, after each match day and store it on a server. All questions with game span entity *this season* could then be retrieved faster. For other more specific questions there would still be a run time of 20-30 seconds. This time could be reduced through optimizing the code. However, most of the time comes from retrieving data from the Twelve website, which can not be made faster from the bots perspective.

In Section 4.2 one can see that some errors occurred for the chat bot. In Figure 6 the rendering have not worked and a backup screen shot of the websites homepage has been used. This problem mainly occurs because Twelve's web page only displays the 50 players with highest number of total points. If the question was about the worst player in an entire league there is a risk that the player is not in the top 50 on total points. The chosen player does not have a pitch visualization and thus the bot cannot take a screen shot. This is not a big problem because it only occurs on questions about worst player in an entire league. It is also possible to solve, by doing a new request from twelve with only that player. This would make the response time longer, but it would make sure that the message always have a pitch preview.

In Figure 8 the bot could not answer the question and responded with an

error message. This was because Wit was not able to find the contact(player name) Lindelöf. Wit has problems understanding words with letters not in the English alphabet. Contact is an entity everyone that uses Wit can use, and thus it is being trained by everyone. This means that the training we do with contact will have little impact, meaning this issue can not easily be solved by simply training Wit. However, using contact has had its advantages. If it was not able to pick up players by free text we would have had to create an entity for player names which would be a list of all players, playing in the Premier League and Champions League. One would constantly need to update this list manually because teams get new players and this would not be practical.

Another issue with the contact entity is that it have problems with names that also are words in English. This can make it miss names but also find names that are not supposed to be names. For example, if someone is asking about the player Hazard it can instead recognize it as a word for danger. The opposite is also true when asking about who the best player is. Wit sometimes recognizes best as a name, since it is a surname in English.

At the moment, the bot answers the questions in a similar tone no matter how the questions are asked. To get the feeling that you are actually having a conversation with a real football expert, one could implement the use of sentiment. The sentiment explains if there exists a positive, negative or neutral tone in the underlying message. If a user asks *who is the worst player in Premier League?* or *why is Salah so great?* it would be interesting if the bot answered in a negative or positive way depending on what sentiment is detected. This would lead to the bot feeling more alive and less repetitive.

6 Conclusions

A bot that creates correct, interesting tweets and answer questions has been developed and thus, the goals of the project has been accomplished. We have increased the number of things the bot can talk about and implemented so that it chooses to talk about things based on how interesting they are. The chat bot can answer questions with some of the more common intents about players in an interesting way. The bot have a few issues with interpreting messages correctly and creating sentences, but overall the bot works well.

6.1 Future work

To proceed forward, we suggest that the bot should be launched on another platform to decrease the time for answering questions.

Now that the bot has a clear structure, it is easy to implement more things the users can ask about without having to do major changes to the code. Examples for things to implement is more leagues from different nations and add the possibility to ask about player positions like *who is the*

best goalkeeper?. To increase the variation of answers, the use of sentiment could be developed. This would mean that depending on the detected sentiment in the question, the bot would slightly change the wording of the answer. A final thing would be to implement more intents, that the user can ask about. These intents could be chosen by checking frequently asked questions the bot cannot answer.

References

[1] Chu Z., Gianvecchio S., Wang H., Jajodia S. (2012) *Detecting Automation of Twitter Accounts: Are You a Human, Bot or Cyborg?*. IEEE Transactions on Dependable and Secure Computing. 9 (6): 811-824.

[2] Srinivasan K., Nguyen C., Tanguturi P. (2018) *Chatbots Are Here To Stay* , Accenture 2018

[3] Brandon J., *Whole Foods just launched a Messenger chatbot for finding recipes with emojis* , https://venturebeat.com/2016/07/12/whole-foods-just-launched-a-messenger-chatbot-for-finding-recipes-with-emojis/ (Accessed 2019-01-03)

[4] Hern A., *Can Duolingo's chatbot teach you a foreign language?* , https://www.theguardian.com/technology/2016/oct/06/duolingo-chatbots-learning-language (Accessed 2019-01-03)

[5] de Angleis L., *Facebook Messenger: The Best Soccer Bots For 2018* , https://thebotplatform.com/the-best-soccer-bots-on-messenger/ (Accessed 2019-01-03)

[6] *Wit.ai Documentation*, https://wit.ai/docs

[7] E. Bruce Brooks (2005), *Statistics: The Poisson Distribution* , University of Massachusetts at Amherst https://www.umass.edu/wsp/resources/poisson/ (Accessed 2019-01-04)

[8] Leon-Garcia A., third edition, (2008), *Probability, Statistics and Random Processes for Electrical Engineering* , Pearson/Prentice Hall

[9] *Formel- och tabellsamling för grundkursen Sannolikhet och Statistik*, (2015), Uppsala University, Matematiska Institutionen