Group Project ICS2205
Web Intelligence

Charlie Abela, Joel Azzopardi

December 14, 2022

This document contains the details for the group ICS 2205 project which is marked out of 100%, however it is equivalent to 60% of the total mark for this unit. Each team should be composed of a maximum of TWO individuals per team. You can decide with whom you want to team up. However, if you do not manage to team up yourselves, then we will form the teams, and such team setups will be final.

While discussions between individual students are considered as healthy, the final deliverable needs to be that produced by your team and not plagiarised in any way. The work within each team has to be distributed fairly, and in the documentation you will need to describe how the work was distributed and who was responsible for which part of the project. The mark given to each team member is determined based on the quality of that member's contribution to the team's overall project. Marks assigned to different members of the same team *may* vary.

The **deadline** for this project is **12:00pm Friday 10th February 2023**. Deliverables and attached plagiarism form must be uploaded on the VLE. Projects submitted late will be penalised or may not be accepted.

1 Specifications

This project consists of **TWO** tasks both of which involve the analysis of a publicly available dataset. Each task is assigned 50% of the mark for this project. You are to address **BOTH** tasks and will need to create a Jupyter notebook[1] for each task. The notebooks should also contain explanations about any feature of your solution as well as inline comments and markdown detailing important aspects of the code. The notebooks will be considered as the main documentation documents. In the following sections we provide the details for each of the two tasks.

[1] `https://jupyter.org/`

1.1 Task 1: Graph Analysis

The aim of this task is to analyse and visualise a football passing network using Python. A passing network consists of nodes that represent the players on the football pitch and edges that represent the interaction (passes) between the players (see Figure 1). All this happens within a spatial and temporal frame, whereby the pitch is the spatial frame and the duration of one game is the temporal frame.

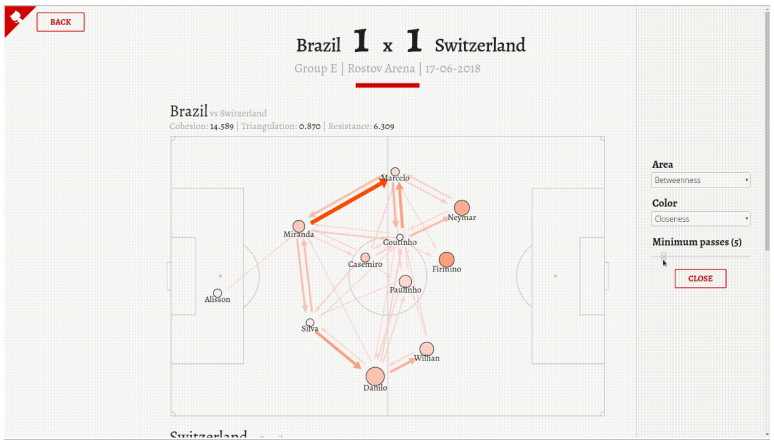

Figure 1: Passing network of a World Cup 2018 game[2]

In this task you will be using the data from the StatsBomb Open Data repository[3]. The data will however be made available to you on Google Drive (since it is over 400MB when zipped) should you have any issue. The data is in JSON format and the related specifications are found in the StatsBomb Data Specification file that is also found on the VLE. You need to consider mainly the specifications related to the events attributes that are important to create the passing network. Nevertheless, you will also need to consider the competitions and the matches files so that you'll be able to select amongst the various competitions and leagues and then select the game/s that you want to analyse.

More specifically, the events data includes information about different types of events that occurred during a match. Each file (e.g. 68348.json) has event information about a particular game. In this case it's a La Liga (Spanish league) game between Barcelona and Espanyol. Each file includes also (at the beginning) information about the teams' lineups (11 players per team: their ids, names and roles). Passing information includes information such as: timestamp, possession team, player that initiates the pass, the location of the player, who is the recipient player, the length of the pass etc.

[3] `https://github.com/statsbomb/open-data`

Your task involves the following (marks are allocated out of 100%):

1. Create a Jupyter Notebook in which you create the passing network from the StatsBomb data for specific matches that are part of a particular competition of your choice (**Total marks: 30**)

   - parse the competition files to get the list of competitions (e.g. Champions League etc) and then for a particular competition get the relevant matches for a specific season. (**5 marks**)
   - from the match id you will need to load the events details for that match, including the lineup for each team and then generate the passing networks. To do this you need to:
     - extract the lineup for each team;
     - then filter the data to extract the passes;
     - get the number of times that some player A passes the ball to some player B;
     - retrieve the top 11 players for each team based on the playing minutes (since players might be substituted during the game);
     - generate and plot the passing network using networkX[4].

     (**20 marks**)
   - finally store the passing network for that match in Neo4j Desktop[5]. It is recommended that you use libraries such as nxneo4j[6] or py2neo[7] to interface with Neo4j.(**5 marks**)

2. Analyse the underlying structure of the passing network by computing the following statistics for the passing network **for each team** using networkX (**Total marks: 20**):

   - total number of passes (**4 marks**)
   - degree distribution (plot the distribution) (**6 marks**)
   - average path length (**5 marks**)
   - global clustering coefficient (**5 marks**)

   In the jupyter notebook provide information about any challenges that you might have encountered when computing these statistics, and explain how you solved them.

[3] `https://github.com/rodmoioliveira/football-graphs`
[4] `https://networkx.org/`
[5] `https://neo4j.com`
[6] `https://github.com/ybaktir/networkx-neo4j`
[7] `https://py2neo.org/`

3. Query Neo4j directly using Cypher to answer the following questions regarding each team (**Total marks: 40**):

   - who is the most active player in terms of passes? (**10 marks**)
   - which players have an intermediary role (list the top 3 per team)? (**10 marks**)
   - how central is a player? (**10 marks**)
   - who is the player that received the highest number of passes? (**10 marks**)

   You need to include the Cypher queries in the jupyter notebook as markdown. Furthermore, you need to explain any parameters used and any encountered issues and how these were solved.

Task 1 is allocated **50% of the total marks**.

1.2 Task 2: Text Analysis

For this deliverable, you are expected to perform text analysis over a News Category Dataset available on Kaggle dataset[8]. The original dataset contains 210,294 news headlines from HuffPost[9]. However, for the scope of this project, this dataset has been filtered to contain a bit over 47K news headlines covering 1/1/2017 till 23/9/2022. This dataset is being made available to you on VLE as JSON records.

For this task you need to do the following:

1. Provide a jupyter notebook that performs the following text analysis tasks on this dataset (**Total marks: 60**):

   a. Process the news headline text: (**14 marks**)
      i. Parse the JSON files, and extract the text from each record (extract the data in "headline" and "short_description".
      ii. Perform lexical analyses to extract the separate words, and to fold them all to lower case.
      iii. Use a standard stop-word list for English to filter out the stop words.
      iv. Use an implementation of Porter's stemmer to reduce terms to their stems (note that you may find a ready-made implementation provided that you reference its source).

   b. Calculate term weights using TF.IDF. Each headline record should be considered as a single document. (**8 marks**)

[8] `https://www.kaggle.com/datasets/rmisra/news-category-dataset`
[9] `https://www.huffingtonpost.com/`

c. Extract the highest-weighted $n\%$ of the terms for each headline category (each JSON record has a field called "category"). (**14 marks**)
   This can be split as follows:
   i. Calculate the average term weight for all terms over the documents within each category.
   ii. Get the highest-weighted $n\%$ of the terms for each category. This list of terms, and their corresponding weights will subsequently be used to build a category keyword-cloud. This keyword-cloud will show what concepts each category generally mentions. n can be determined arbitrarily so that the keyword-cloud will contain neither too much nor too few words.
   iii. Extract the details of each category (including the category name, the list of articles in it, and list of highest-weighted terms for each category) as JSON. This will be used in the visualisation application described below.

d. Use the document vectors to cluster the news headlines using the *k-means* algorithm. The choice of k is up to you. Note that you only need to do a **single** level of clustering, that is, no hierarchies are being requested. (**12 marks**)

e. Extract the highest-weighted $n\%$ of the terms for each cluster generated in the previous step. (**12 marks**)
   This can be split as follows:
   i. Calculate the average term weight for all terms over the documents within each cluster.
   ii. Get the highest-weighted $n\%$ of the terms for each cluster. This list of terms, and their corresponding weights will subsequently be used to build a cluster keyword-cloud. This keyword-cloud will show what concepts each cluster generally mentions. n can be determined arbitrarily so that the keyword-cloud will contain neither too much nor too few words.
   iii. Extract the details of each cluster (the cluster ID, the list of articles in it, and list of highest-weighted terms for each cluster) as JSON. This will be used in the visualisation application described below.

NOTE: You can use NLTK or other ready made tools to handle part a). However, you are expected to implement your own *TF.IDF* weighting scheme, *Cosine Similarity* measure and *k-means* clustering. Note that you can re-use any of the parts implemented for the Text Analyses Task in the Individual Project in this task as well.

2. Set up a simple web application that can visualise the results obtained. Namely, you need to (**Total marks: 40**):

   a. Set up a simple web application using Flask. (**10 marks**)
   This involves the following:

      i. Set up Flask.
      ii. Import the JSON files generated in the Text Analysis part.
      iii. Show the list of documents as a 'clickable' list.

   b. When a document is clicked, its details (headline, date, etc) and the corresponding keyword cloud should be shown. (**10 marks**)

   c. Show the list of categories as an interactive bubble chart. When a category bubble is clicked, the list of documents which who form part of that category should be shown together with the keyword cloud for that category. (**10 marks**)

   d. Show the terms' clusters as an interactive bubble chart (similar to point *c* above). When a cluster bubble is clicked, the list of documents which who form part of that category should be shown together with the keyword cloud for that cluster should be shown. (**10 marks**)

   IMPORTANT: You need to include the necessary instructions on how to start the Flask application and import any necessary data in the Jupyter notebook as markdown. You should also include information on what sources did you use for your visualisations, and describe any challenges encountered.

   NOTE: Word cloud visualisations should be generated using any available JavaScript library. Some examples are found in: `https://github.com/wvengen/d3-wordcloud` and `https://github.com/shprink/d3js-wordcloud/`.

   NOTE 2: Some helpful resources to display a bubble chart on your web page: Bubble chart[10].

Task 2 is allocated **50% of the total marks**.

1.3 Summary of deliverables

Task 1:	Graph Analysis and Visualisation	50 marks
Task 2:	Text Analysis and Visualisation	50 marks

You will need to submit the following on the VLE:

[10] `https://www.webtips.dev/how-to-make-interactive-bubble-charts-in-d3-js`

i. The relevant files (zipped) associated with Task 1. Include also a plagiarism form, duly filled-in;

ii. A PDF export of the Jupyter notebook associated with Task 1 in the relevant *TurnItIn* area.

iii. The relevant files (zipped) associated with Task 2. Include also a plagiarism form, duly filled-in;

iv. A PDF export of the Jupyter notebook associated with Task 2 in the relevant *TurnItIn* area.

2 Final Remarks

Final suggestion: if you have difficulties do not hesitate to contact us. Any issues – including technical difficulties, or difficulties between team members – should be identified and highlighted as early as possible to ensure timely resolution.

Good luck!!