Building a Social Recommender System

By

TEAM 11

Akhil Raveendran
Deepika Yannamani
Rajesh Thummala
Sheshank Makkapati
Siddarth Varanasi

Table of Contents:

1. Abstract

The goal of the project is to build a social interest recommender system.This is done by associating a set of interests extracted from the user activities.Also done by extracting interests from their friendship list (by selecting those friends in the list that indicate an interest more than a peer friendship relation).We generated clusters of similar users (i.e. with similar interests) – using user's social network relationships.We evaluated the clusters using analysis algorithms. By mapping user's data and their interests, we recommend social interests for each user.

Goals of the Project:

- Suggesting different interest categories to new users, without any previous user activity to base the recommendations upon.
- Second is to rank the existing user's new recommendations based upon his recent activity.

Suggesting different interest categories to new users, without any previous user activity was done using Collaborative and Content Based filtering.Ranking the existing user's new recommendations is achieved using Knowledge based approach.In both the solutions, Kmeans clustering was used to create category clusters and similar users clusters.The dataset used is obtained from Twitter API through which the information of the users is gathered and then mapped on to the Wikipedia.

A **Collaborative approach** is used first. We collect the community data, 425145 users each one with dozens of interests (which will be shown later in the data preparation section) among 216686 possible interests. However data is extremely sparse, every interest carries few information (it is only a 0/1, not a rating on 5 or 10) and does not make use of our knowledge

engineering, the wikipedia embeddings.

Knowledge based approach has proved to be very powerful in the categorization task, where with "powerful" we refer to the high semanticity of the distance among embeddings, even after a dimensionality reduction. An item based recommender system of new generation is built using the embeddings, where the similarity between two items is simply given by the **cosine similarity** between the corresponding vectors. This approach is a clear step with respect to the classic item based approach, embeddings encoding of relevant information about the item is enormously more effective and efficient than the standard hand crafted keywords. We have chosen this solution.

2.Introduction

Because of the critical advancement of Web 2.0, users can produce or share data on networks and participate in different cooperations with other users. For instance, when shoppers buy items through internet business locales or physical stores, they can impart data about their encounters to different purchasers by means of reviews on websites. At the end of the day, this tendency empowers users to effectively obtain important data on items by means of the Internet; it likewise makes users increasingly dependent on online data with respect to buying patterns.

Moreover, a few works have proposed an assortment of recommender systems. The fundamental component of a recommender system includes deciding user inclinations by investigating a lot of crude information, by sifting through raw data to locate the important data identified with potential user inclinations. For the most part, recommender systems work through the collaborative filtering approach and content-based filtering approach. The collaborative filtering approach (CF) is the most used method in many domains; it can be classified as user-based CF and item based CF.

For the above reasons, our project proposes a recommendation method which combines

collaborative filtering, content based and knowledge based approach to make predictions for users in the social networking environment.

3.Related Work

There are a variety of related projects which give summary of a platform to capture opinions related to places and generate a positive or a negative recommendation for users based on both the global use of Twitter as well as the comments made by trusted people on the network using Interaction Analysis and Trust Analysis.

But video technology is used in the Interaction Analysis which makes it less reliable and also collection of data will be a daunting task. So, we have used the Collaborative and content based filtering methods which are conventional methods for building a recommendation system.

4.Exploratory Data Analysis

The first and primary step that we do after receiving any data set is the exploratory data Analysis. In this step we first look at the data and then clean it and check for any important visualizations and prepare the data by changing it into a tidy format so that we can model it perfectly and obtain good results.

Data Preparation:

We have one dataset for this and it is as follows:

- Wiki-MID_EN.nt.gz

A very large Multi domain interests Dataset of Twitter users with Mapping to Wikipedia is used. We are using this for training and testing as well. The dataset includes number of multidomain preferences per user on music,sport,books,movies,politics etc.The preferences are extracted from sources such as the messages of users who use Spotify, Goodreads and other content sharing Platforms or from Topical friends.This extraction is done in 3 steps:

- Extracting the interests from users' textual activities or communications.
- Extracting interests from users' friendship lists.
- Mapping interests on to Wikipedia pages.

Extracting interests from users' textual communications:The natural way for modeling users' interests is done using textual features extracted from users' communications, profiles or lists. However, When applied to large data streams,this information source has several drawbacks such as the set of Twitterusers. First, it is computationally very difficult to process millions of daily tweets in real time; secondly, the extraction process is error prone, because of highly ungrammatical nature of micro-blogs.

Everyday a large number of people use online platforms such as YELP,spotify etc which are commonly known as content sharing platforms.

Spotify: Spotify is a music service offering streaming of music which is on demand for both the desktop and mobile users. Users can also create playlists, share and edit them with other users. In addition to accessing the Spotify website, users can retrieve additional information such as the record label, song releases, date of release etc.

The standard form of these tweets is: #NowPlaying <title>by <artist > <URL>

IMDb and TVShowTime: In the domain of movies, currently there are no dominant services. Popular platforms in this area are Flixter, themoviedb.org and iCheckMovies.

Extracting interests from users' friendship lists: This information is available in users' profiles

and does not require additional textual processing. Furthermore, interests inferred from topical friends are less volatile since, ”common” users tend to be rather stable in their relationships. Topical friends are therefore both relatively stable and readily accessible indicators of a user’s interest. The advantage in this process is that average Twitter users have hundreds of followees, many of which, are indicators of a variety of interests in different domains,rather than genuine friends in fields like entertainment, sport, art and culture, politics, etc.

Mapping interests onto Wikipedia pages: The final step is to associate each interest, either extracted from messages or inferred from friendship relations,with a corresponding Wikipedia page, e.g.:

@bbc ⇒WIKI:EN:bbc

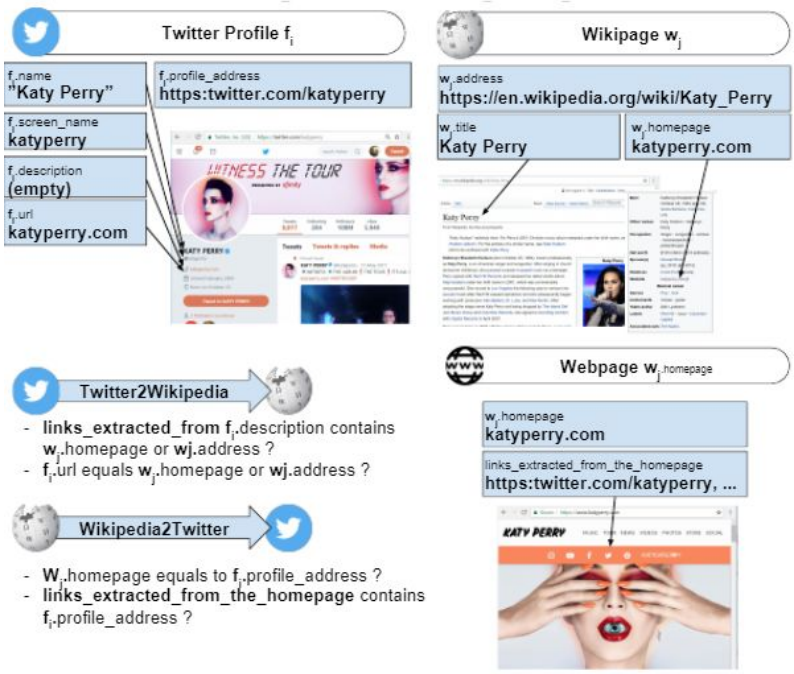


Fig: Example of Twitter2Wikipedia and Wikipedia2Twitter Mapping.

5.Loading dataset and Pre-Analysis

The dataset has to be loaded in the convenient form i.e Dictionary. These are implemented using hash tables as they are loaded as Dictionaries. During the data is being loaded the data has to be analysed. This analysis will turn useful when choosing the most suitable method for data analysis.

Analysing type of relations: The type of relations such as follows, likes,related match and Type are analysed. After the data is loaded, it is plotted and the following plot is obtained.

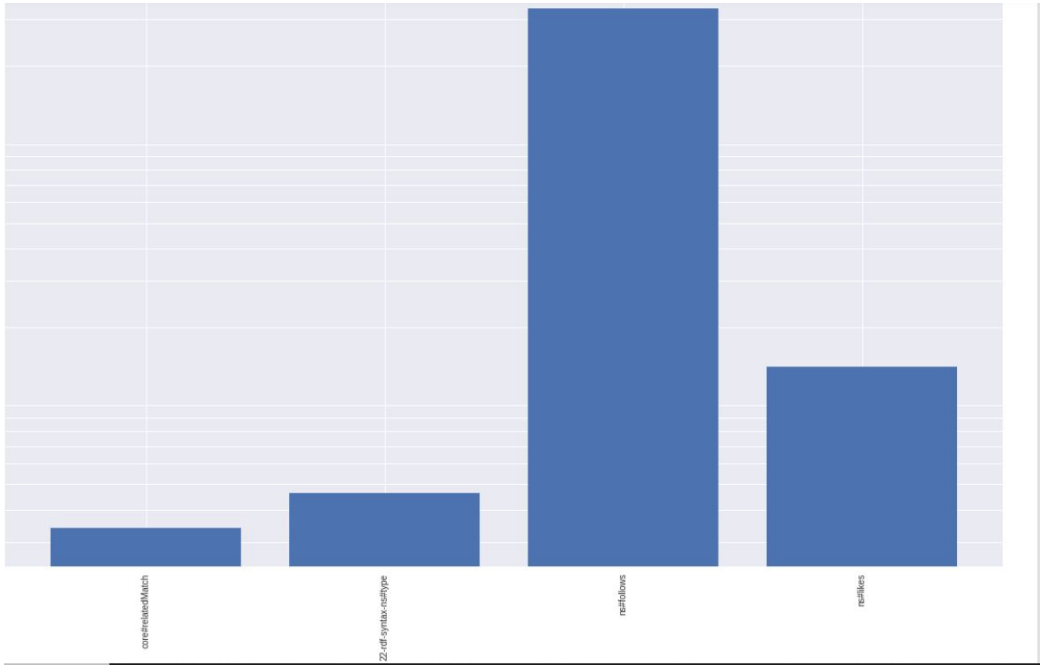


Fig: The plot of the type of relations in Data.

Analysing the type of Subjects: The subject here refers to the first element of every triple, ignoring the relation type. e.g. 'https://play.spotify.com/user/', 'https://twitter.com/intent/', 'https://www.goodreads.com/book/show/'


```
{'https://': 1,
 'https://goodreads.com/review/show/': 10236,
 'https://open.spotify.com/album/': 12808,
 'https://open.spotify.com/artist/': 1760,
 'https://open.spotify.com/concert/': 5,
 'https://open.spotify.com/episode/': 29,
 'https://open.spotify.com/show/': 25,
 'https://open.spotify.com/track/': 73661,
 'https://open.spotify.com/user/': 4008,
 'https://play.spotify.com/album/': 7,
 'https://play.spotify.com/artist/': 2,
 'https://play.spotify.com/track/': 63,
 'https://play.spotify.com/user/': 12,
 'https://spoti.fi/': 160926,
 'https://twitter.com/intent/': 35002172,
 'https://www.goodreads.com/book/show/': 69,
 'https://www.goodreads.com/review/': 6879,
 'https://www.goodreads.com/review/show/': 3526,
 'https://www.imdb.com/title/': 8282}
```


Fig: All the subjects along with their Occurences.

Building dataset in Dictionaries: The name of every dictionary follows the standard key_value.

e.g. interest_wiki,

interest_interesttypeid,

user_interests,

user_wikipages,

numberofinterests_occurrences.

Compute data: We need a function that extracts unique Id from every link. So, scraping is done from the links.

```
def id_scraper(url_string):
  for  counter, character in enumerate(reversed(url_string)):
    if character == '/' or character == '=':
      return url_string[-counter:-1]

print(id_scraper('<https://twitter.com/intent/user?user_id=1483528441>'))
print(id_scraper('<https://www.goodreads.com/book/show/22856899-the-tribute-bride>'))
```

```
1483528441
22856899-the-tribute-bride
```

Fig: scraper code snippet.

After this scraping is done the interest_wiki, interest_interesttypeid, user_interests are built.With user_interests we can now build user_wikipages and numberofinterests_occurrences. Finally the desired data is obtained in the Dictionary format.

6. Categories from interests

Extracting categories to synthesize interests using semantic is not a well defined problem, how can we measure the performance of our approach? We should make some assumptions:

1. Each interest should be semantically related to its category
2. Categorization should be balanced
3. Granularity should be appropriate: too many categories are redundant, too few categories weaken the model due to the loss of information.

The balance of the categorization is an easy measure but what about the other two criteria? Let's see our approach to the problem.

We used precomputed wikipage embeddings as semantic model, these are obtained applying Wor2vec algorithm to reading sessions (sequence of articles visited by a user during a wikipedia session), where articles that tend to be read in close succession have similar representations. Since people usually generate sequences of semantically related articles while reading, these embeddings also capture semantic similarity between articles. Each vector has 100 dimensions.

We select all vectors corresponding to wikipedia articles present in the WikiMID dataset (216.000 out of 1.828.000) and perform a clusterization using MiniBatch K-Means, Minibatch because of the size of the dataset, K-Means because it is more prone to generate balanced

clusters and the number of clusters is tunable. However clustering suffers from the curse of dimensionality and 100 dimensions are too much, so we performed at first a dimensionality reduction with PCA, the final number of dimensions is chosen looking at the explained variance, 6 dimensions in our case.

The number of clusters is decided looking both at the silhouette coefficient (look below for more details) and at the semantic range of categories (we do not want too few clusters even if the corresponding silhouette coefficient is high, all the categories would be about music).

The final clusters represents the categories, we named each category looking at the centroid of the cluster, the name follows the standard: *Macro Category (Centroid page name)*. Where Macrocategory refers to the category of the centroid page in Babelnet (e.g. Music, Literature and Theatre...). The simple Babelnet categorization is not good because led to a heavy unbalanced categorization, for instance Music would include more than 60% of all the pages. With our method instead we have many music categories, distinguished by the centroid page name, e.g. Music (Blind_Melon), Music (Yo_Gotti)... This partition of the categories is meaningful, for instance Music (Blind_Melon) can be associated to USA Rock, while Music (Yo_Gotti) to Rap.

Importing Wiki Page embeddings

Wiki Page embeddings are managed with the WikiEmbedding class as suggested by the official documentation.

WikiEmbedding class implements two methods:

most_similar returns the n most similar wiki pages to a given one using cosine similarity

wiki_vector returns the vector corresponding to a given wiki page (implemented by us)

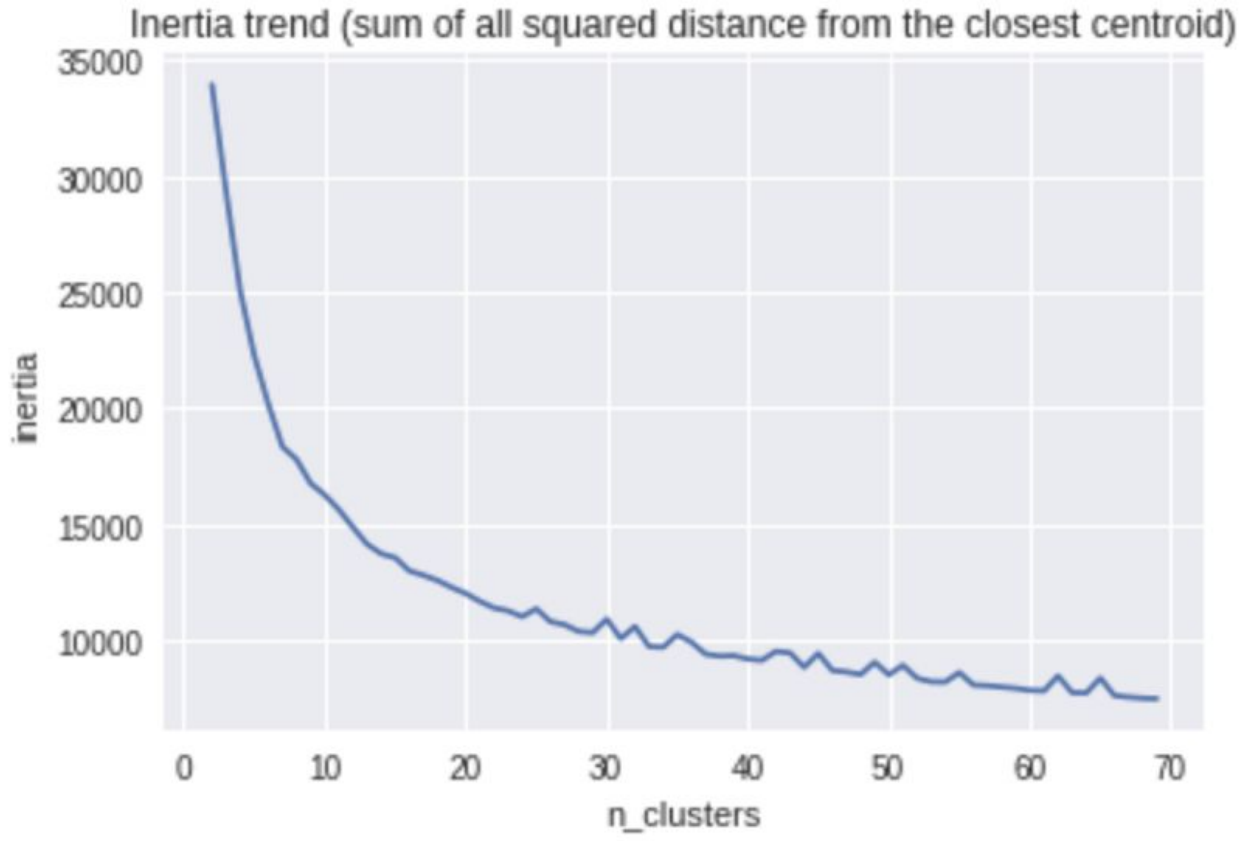
Building wiki embeddings of wiki pages representing interests

Selection of the Wiki Pages corresponding to an interest. A new WikiEmbedding object is created.Some pages are not found, the problem is especially with pages that need a disambiguation. WikiMID dataset and official Wiki Page embeddings make use of a different

naming standard for those pages.Fraction of the pages not found and the final size of the original wiki embedding compared with the selection of only the interests wiki embedding. Reducing dimensionality with PCA, we choose 6 dimensions after some tuning.

How many clusters? Let's see manually meaningfulness and granularity of results and take a look to the Silhouette coefficient trend for a range of cluster from 2 to 70. Silhouette coefficient is explained later.

Inertia (sum of all squared distance from the closest centroid) decreases naturally as the number of cluster increases. It is worth to look at the inertia trend to check any second order singularities, those may correspond to a good choice for the number of clusters. In our case however the trend is smooth without any "elbow".



The Silhouette trend is more interesting.

The Silhouette coefficient is a measure to evaluate clustering that does not need ground truth labels, it is based on how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

We used sklearn implementation of the silhouette score defined as follows:

The Silhouette Coefficient is defined for each sample and is composed of two scores:

- **a**: The mean distance between a sample and all other points in the same class.
- **b**: The mean distance between a sample and all other points in the next nearest cluster.

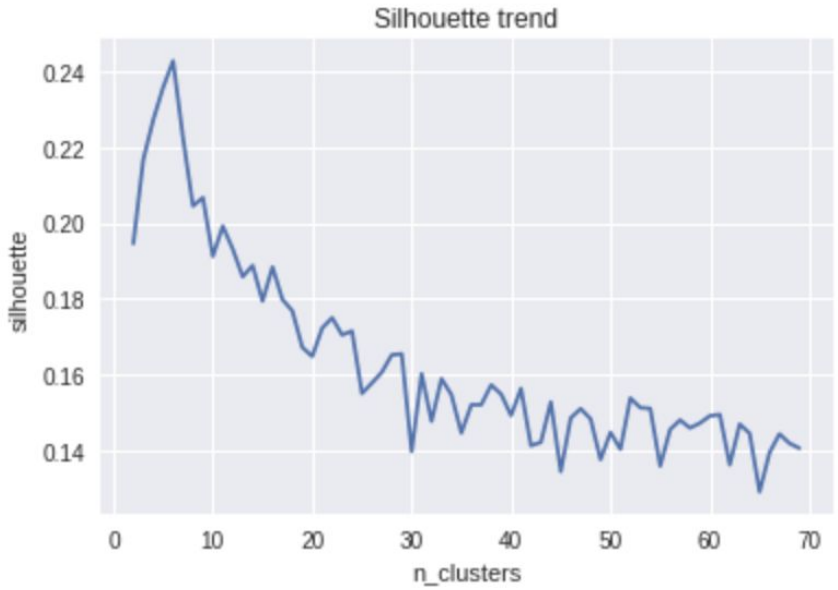
The Silhouette Coefficient *s* for a single sample is then given as:

$$s=(b-a)/max(a,b)$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.

The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

We used euclidean distance.



Despite few clusters guarantee a higher Silhouette coefficient we choose 41 clusters that provide a sufficient granularity and is near a local maximum of the Silhouette trend.

Now we have to name the categories, as described above, we make use of the Babelnet domains. This is accomplished using two files, a sample of each one is shown below.

As said before the rough categorization with babelnet domains led to poor results due to the heavy unbalance with our dataset. And the high number of missing wiki pages.

We used two official sources of Babelnet:

babeldomains_wiki.txt with the category associated to each wikipage and the corresponding confidence score.

2010 Proton Malaysian Open – Singles Sport and recreation 1.0

Promnik, Masovian Voivodeship Geography and places 1.0

Mark Rankin (record engineer) Music 1.0

Conformal film Chemistry and mineralogy 1.0

domain_list.txt with all the domains (categories) listed

Animals
Art, architecture, and archaeology
Biology
Business, economics, and finance
Chemistry and mineralogy
Computing
Culture and society
Education
Engineering and technology
Farming
Food and drink
Games and video games
Geography and places
Geology and geophysics
Health and medicine
Heraldry, honors, and vexillology
History
Language and linguistics
Law and crime
Literature and theatre
Mathematics
Media
Meteorology
Music
Numismatics and currencies
Philosophy and psychology
Physics and astronomy
Politics and government
Religion, mysticism and mythology
Royalty and nobility
Sport and recreation
Textile and clothing
Transport and travel
Warfare and defense

Finally we have the categories with their human understandable names. Since not all wiki pages are present in *babeldomains_wiki.txt*, if the centroid is missing we deduce the macro category from the first present close neighbor.

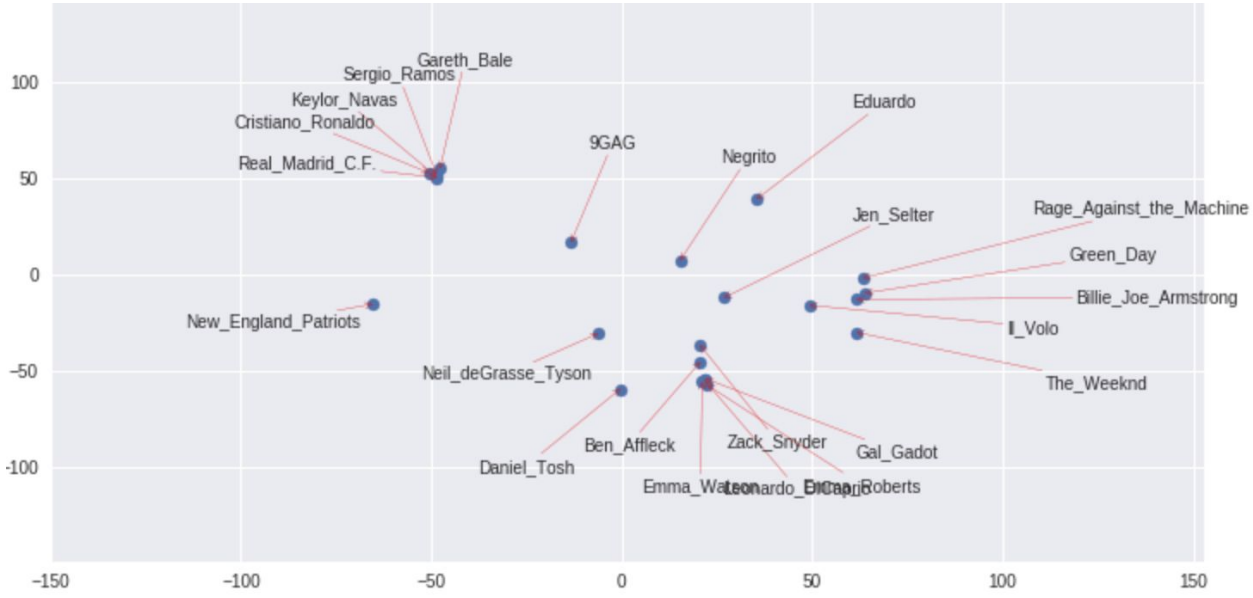
Example of a user

In this subsection we tried our categorization method with a single user to have feedback. Below are shown the processing and the results.

TSNE plot code

With a single user we can plot the wiki page vectors corresponding to interests, in this way we can see if the distance between them has a semantic meaning. So we can have an idea of the performance of the chosen semantic model.

Our plot should be in two dimensions, so we use TSNE algorithm after PCA to reduce further the dimensionality. TSNE is the standard for this type of data visualization.



This is the plot. Results are very good, we can easily distinguish three clusters: Sport, Cinema, Music. Even the location of elements in a single cluster appears strongly semantically meaningful. Look at the Sport cluster, all instances related to Real Madrid are very close while New England Patriots are more distant.

Below are shown the elements most similar to Cristano Ronaldo inside the user interests.

[('Gareth_Bale', 0.861935873465163),

('Sergio_Ramos', 0.8490769847592015),

('Real_Madrid_C.F.', 0.823372014072481),

('Keylor_Navas', 0.769464261036644),

('Eduardo', 0.512091795320242),

('9GAG', 0.4643334584321524),

('Leonardo_DiCaprio', 0.4310935926104691),

('Jen_Selter', 0.4310021568355299),

('The_Weeknd', 0.4180912948597736)]

This is the final categorization. Look how all the Real Madrid related elements and New England Patriots are in two different Sport and Recreation categories. One concerning football (Darrell Curie) and the other american football (Tyson_Chandler_(American_football)).

A final test for the semantic model

A final test for our semantic model, can we do simple operations like the classical word2vec?

In the classical word2vec we have:

$$mostsimilar(KING - MEN + WOMAN) = QUEEN$$

The answer is yes:

$$mostsimilar(CRISTIANORONALDO - REALMADRID + ACMILAN) = FILIPPOINZAGHI$$

7. Generating clusters of similar users

At this point each user is represented by a 41 dimensional vector where each component represents the occurrences of the corresponding category.

Which community detection method should we use with this data?

We do not have a graph representation of our users, we could generate a simple graph representation using for instance cosine similarity (not sensible to the unbalanced norm of our user-vectors) and putting a similarity threshold to have a link between two nodes (users). However this is not ideal for the high computational cost due to the size of our dataset and the loss of information. So we discard community detection methods based on a graph representation.

So we look for a classical clustering method. Our attention is focused on two clustering algorithms: DBSCAN that is a good choice for non-flat geometry and can use cosine similarity and the classical KMeans that can work effectively on big datasets in its Minibatch version and gives more guarantees on the cluster's balancing. See here for more details on the clustering algorithms (sklearn user guide).

Finally we decided newly for Minibatch Kmeans after some experiments.

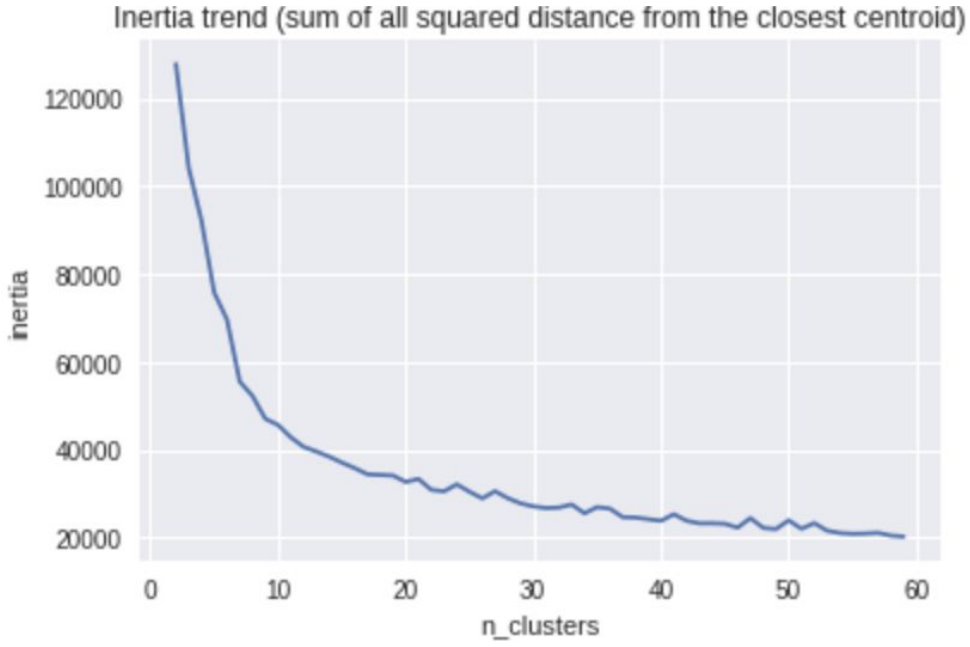
Kmeans with euclidean distance

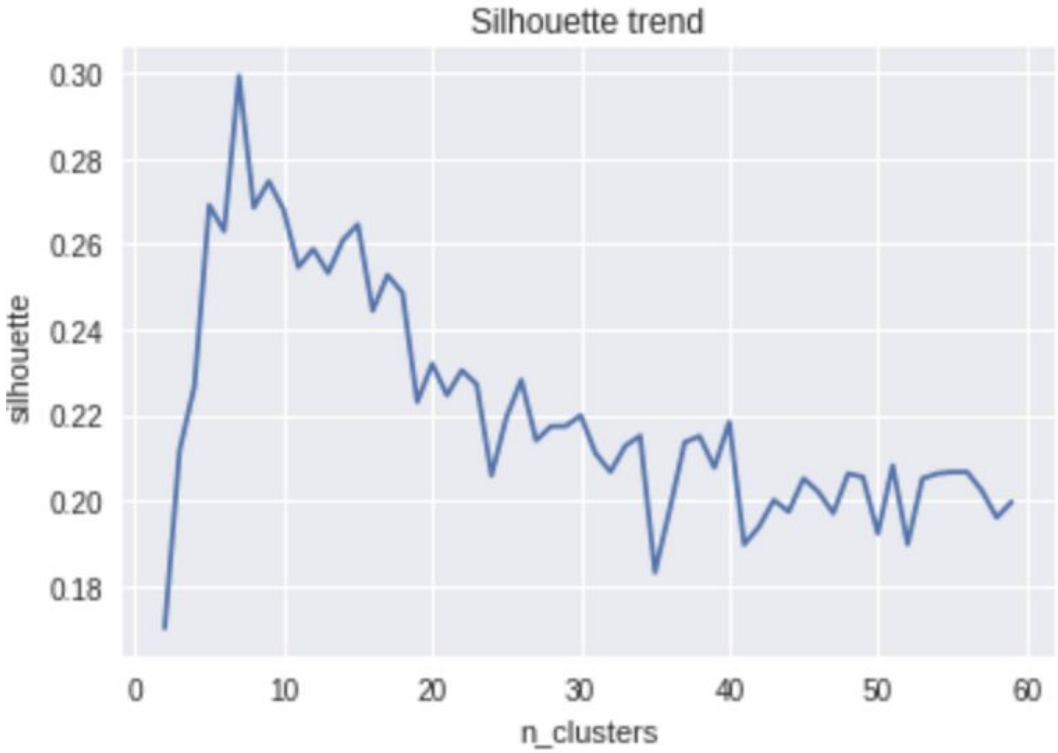
Kmeans requires some preprocessing in order to properly use the euclidean distance, that becomes almost meaningless in high dimensional spaces and with heavy unbalanced norms as our dataset.

So we perform a normalization and a PCA dimension reduction.

Note that using different metrics in Kmeans is not correct, it may stop converging with other distance functions. There exist algorithms based on the Kmeans idea that work with other distance function (e.g. Kmedians for Manhattan distance). Here we choose to use the classical Kmeans after a dimension reduction.

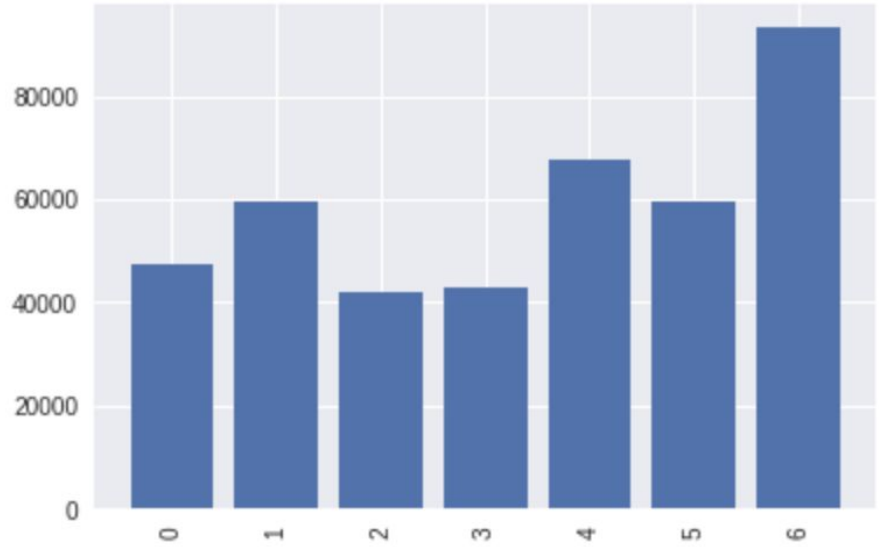
How many clusters should we have?.....To find out that we have to plot Inertia trend and Silhouette score again.





We choose 7 clusters looking at the maximum of the silhouette trend. A higher number of clusters can be selected if a higher granularity resolution is desired .

It follows the final clustering.



8. Associating twitter users to the appropriate cluster of users

Using twitter API we obtained friendship information of the new users in S21.tsv. We stored this information in the twuser_friends dictionary, key: twitter userID, value: list of the IDs of the friends (who is followed by the user).

We can consider this problem as finding the best community (cluster) for each user, a recommendation system where the data available for each user can be considered as a list of peers or as a list of interests.

At this point we have two choices to associate each user to a cluster:

1. **Collaborative approach.** If most of the friends are already tagged with a cluster (i.e. they are in user_cluster dictionary) we can simply assign to each user the cluster corresponding to the most frequent cluster among its friends.
2. **Content based.** If the first method is not applicable, we should look at the friends as interests, collect the corresponding wiki pages, parse in the above categories, normalize the category vectors and predict the cluster of each user using the already trained MiniBatch Kmeans model used in the clustering point.

First of all let's query the twitter API (or load our dictionary if we have built it before).

We used tweepy, a common python library to deal with twitter API.

First we have to find interests of users. First of all we check that a sufficient number of friends, now viewed as interests, has a corresponding wiki page. 50% is not ideal but is sufficient considering that the average number of friends for each user is above 1000.

If we want to check the cluster of a specific user we use below code snippet.

```
twitter_user_id = "100246585" #@param {type:"string"}
print('The user', twitter_user_id, 'has',
      len(twuser_friends[twitter_user_id]),
      'friends,', len(s21user_wikipages[twitter_user_id]),
      'of them correspond to a wikipage.\nThe associated cluster is',
      s21user_cluster[twitter_user_id])

```

Result:

The user 100246585 has 63 friends, 17 of them correspond to a wikipage.

The associated cluster is 4

9. Recommend to the 500 users - 3 out of the 6 proposed items

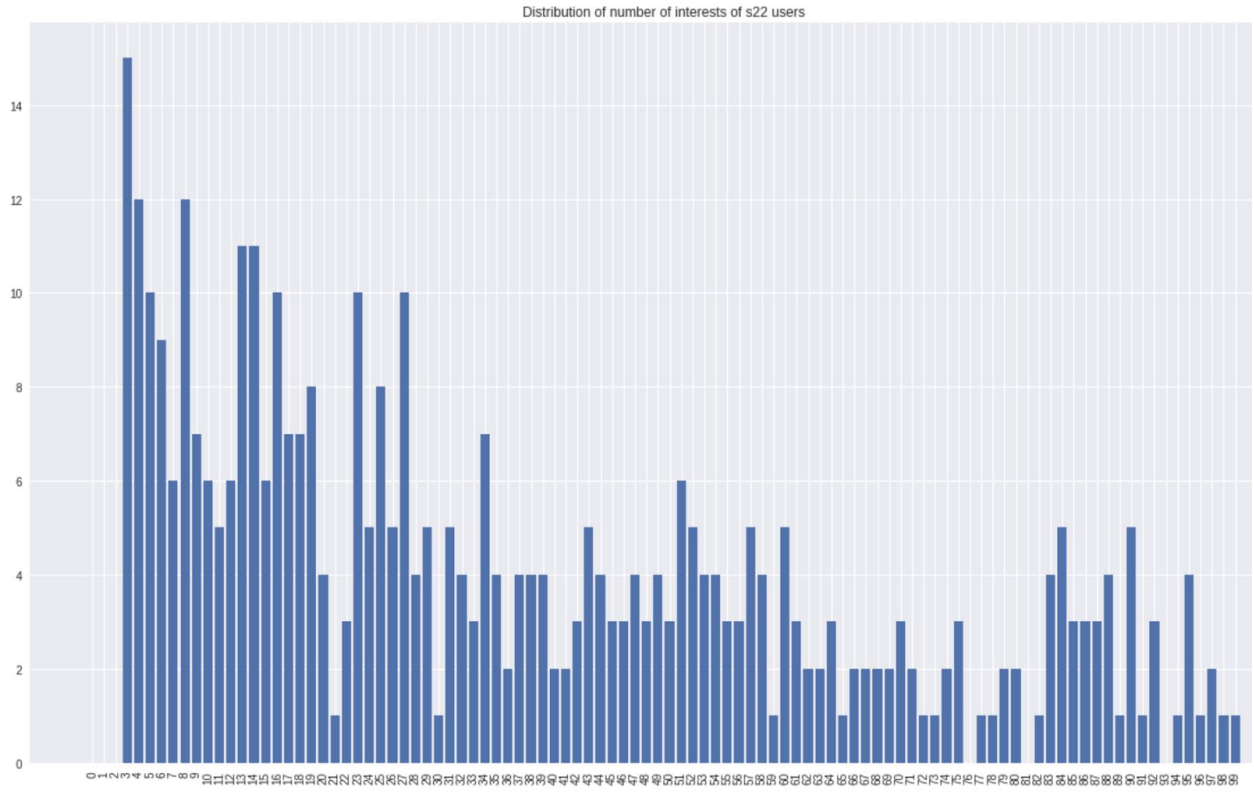
This final task requires the implementation of a recommender system. We have 500 twitter users with a list of their interests and we are requested to decide 3 new relevant interests for the user out of 6 proposed items.

Considering all the work so far, which recommender system shall we use?

- A **Collaborative approach** is possible. We have community data, 425145 users each one with dozens of interests (see distribution in the histogram above in the data preparation section) among 216686 possible interests. However data is extremely sparse, every interest carries few information (it is only a 0/1, not a rating on 5 or 10) and does not make use of our knowledge engineering, the wikipedia embeddings.
- **Knowledge based approach**. Our knowledge engine has proved to be very powerful in the categorization task, where with "powerful" we refer to the high semanticity of the distance among embeddings, even after a dimensionality reduction (look for instance at Example of a user). Using the embeddings we can build an item based recommender system of new generation, where the similarity between two items is simply given by the cosine similarity between the corresponding vectors. This approach is a clear step forward with respect to the classic item based approach, embeddings encoding of

relevant information about the item is enormously more effective and efficient than the standard hand crafted keywords. We have chosen this solution.

First of all we import the known interests for each of the 500 new users.



The only drawback of using the wikipedia embeddings is that some pages are missing, it is only a small percentage of the total but we have to take this problem into consideration. While in the categorization task we could simply ignore those pages, here the risk is to have a missing page among the 6 to be ranked.

It is not very clear which type of page is missing, some are simply present but with little changes in the title name (especially special characters and disambiguation pages), others simply are not present.

Here we studied more in-depth which kind of page is missing, hoping the problem was the little changes in the title name in most of the cases (we could try to fix it) but unfortunately most of the problematic pages are simply missing.

Out of 10 only Citytv_Bogot has a consistent substitute.

We will ignore missing wikipages also in this task.

```
MISSING PAGE: Mix_Up    POSSIBLE SUBSTITUTE: Mix_Master_Mike
MISSING PAGE: OCESA_Seitrack    POSSIBLE SUBSTITUTE: OCLC
MISSING PAGE: Citytv_Bogot♦    POSSIBLE SUBSTITUTE: Citytv_BogotÃ¡
MISSING PAGE: Radioshow         POSSIBLE SUBSTITUTE: Radiosurgery
MISSING PAGE: E3        POSSIBLE SUBSTITUTE: E331_series
MISSING PAGE: Cut_Memey         POSSIBLE SUBSTITUTE: Cut_Me_Some_Slack
MISSING PAGE: Muy_Interesante   POSSIBLE SUBSTITUTE: Muy_Dentro_de_Mi_CorazÃ³n
MISSING PAGE: Jornal_Hoje       POSSIBLE SUBSTITUTE: Jornal_Nacional
MISSING PAGE: Susie_Cagle       POSSIBLE SUBSTITUTE: Susie_Castillo
```

Ranking pages

Here it is performed the ranking, we are going to assign a score to each one of the 6 proposed wiki pages and we will choose the best 3.

Take a user u, indicating with $q1...q6$. q1...q6 its query wiki pages, let Iu,qi ={$iqi1...iqi10$} be the set of the 10 pages more similar to qi among u interest wiki pages.

The score for each page qi $score(qi)=\sum jcossim(qi,iqij)$

Iu,qi may have less than 10 elements if the user u has less than 10 interests or if there are less than 10 interests over the similarity threshold of 0.1The parameters 10(number of interests considered) and 0.1(similarity threshold) were chosen after some tuning and taking in consideration the number of interests per user.We decided not to make a normalization of the scores since we considered informative a lower score due to few elements in Iu,qi. This makes possible a comparison between page scores of different users. We saved the results in the file S24.tsv. A sample of the results is shown in output, it is possible to choose the size of the sample.

10. Conclusions

We have created a social interest recommendation system based on the WIKI_MID dataset and using Wikipedia embeddings as semantic model.

We have completed all tasks with success. Our approach was effective, the only problem was the incompleteness of our data (missing wiki pages both in the WIKI_MID dataset and in the Wikipedia embeddings) that affected particularly the last point.

11.Future Scope

In future scope, user communications can be analyzed to measure the role weights of users in social networks, and the effect of role weights on recommendation methods can be assessed at that point. Moreover, the content of user reviews can be investigated by text mining and semantic analysis methods to precisely recognize user preferences and effectively enhance the performance of recommendations.

12.References:

https://medium.com/topic/social-media

https://www.analyticsvidhya.com/blog/tag/social-network-analysis/

https://www.analyticsvidhya.com/blog/2018/04/introduction-to-graph-theory-network-analysis-python-codes/

https://towardsdatascience.com/impact-of-data-and-analytics-on-social-media-in-2018-595a3bd4fb60

https://towardsdatascience.com/tagged/social-network-analysis

https://towardsdatascience.com/tagged/social-media

https://medium.com/s/story/how-to-fix-what-social-media-has-broken-cb0b2737128

https://www.analyticsvidhya.com/blog/2017/02/social-media-analytics-business/