



M2 NLP

UE 901 EC3: Intelligent Systems and Recommendations

---

# Project Report - Final

---

*Authors:*
Asmaa Demny
Fatima Habib
Cécile Macaire
Chanoudom Prach
Ludivine Robert

February 15, 2021

# Contents

# 1 Introduction

In the context of a job offer from Amazon, we want to build an efficient recommender system which consists mainly of "*filtering information that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications*". This project aims to build a recommender system that will take a Wikipedia page as an input, and will recommend 10 new links based on information that will be defined in this report.

The following report will present the main points:

1. Data analysis and characterization,
2. Concurrence study (similar contexts),
3. Proposition of our recommender system strategy with the libraries,
4. Chosen approaches,
5. Analysis of the whole solution and conclusions.

## 2 Data analysis and characterization

From the ENWIKI DUMP PROGRESS ON 20210101[1] which is a Wikimedia dump service, we decided to take the data from:
`enwiki-20210101-pages-articles-multistream12.xml-p8554860p9172788.bz2`.
The following sections will provide information about the data structure, some statistics and the first ideas to answer the given problem.

### 2.1 Data structure & analysis

The data is a XML file of 166.3 MB. The XML file starts with a <mediawiki> tag which encompasses the metadata. The source language is given with the parameter *xml:lang*, here English. The recommender system will therefore only propose pages in this language.
The header of the file shows the site info in the tag <siteinfo> such as the site name ('Wikipedia'), the database name ('enwiki') and the namespaces. Each wiki page is included into a <page> tag.

```
  <page>
    <title>Colegio de Santa Cruz de Tlatelolco</title>
    <ns>0</ns>
    <id>8554864</id>
    <revision>
      <id>965714004</id>
      <parentid>934400346</parentid>
      <timestamp>2020-07-03T00:06:35Z</timestamp>
      <contributor>
        <username>GreaterPonce665</username>
        <id>30826712</id>
      </contributor>
      <minor />
      <comment>{{start date and age}}</comment>
      <model>wikitext</model>
      <format>text/x-wiki</format>
      <text bytes="16412" xml:space="preserve">{{Infobox university
|name              = Colegio de Santa Cruz de Tlatelolco
|image =
File:Iglesia_de_Santiago_Tlatelolco,_M%C3%A9xico_D.F.,_M%C3%A9xico,_2013-10-16,_DD_38.JPG
|native_name       =
|motto             =
|established       = {{start date and age|1536|1|6}}
|type              = [[Catholic education|Catholic]]
|city              = [[Tlatelolco (Mexico City)|Tlatelolco]], [[Mexico City]]
|country           = [[Mexico]]
|campus            = [[urban area|Urban]]
}}
[[File:Iglesia de Santiago Tlatelolco, México D.F., México, 2013-10-16, DD 31.JPG|thumbnail|
Exterior of the church]]
[[File:Iglesia de Santiago Tlatelolco, México D.F., México, 2013-10-16, DD 46.JPG|thumb|
View of dome from below]]
The '''Colegio de Santa Cruz''' in [[Tlatelolco (Mexico City)|Tlatelolco]], [[Mexico
City]], is the first and oldest European school of [[higher learning]] in the
[[Americas]]&lt;ref&gt;{{cite book|url=https://catalog.hathitrust.org/Record/101392426|
title=The first college in America: Santa Cruz de Tlatelolco.|location=Washington DC|
year=1936|author1=Steck|author2=Francis Borgia}}&lt;/ref&gt; and the first major school of
interpreters and translators in the [[New World]].&lt;ref&gt;{{cite book|chapter-
```

Figure 1: Extract from the XML file of a Wikipedia page.

[1] https://dumps.wikimedia.org/enwiki/20210101/

From Figure 1, the most important information that we find for each Wikipedia page are:

- the title (<title>),
- the ID (<id>),
- the parent ID (<parentid>),
- the contributor (<contributor>),
- the text (<text>) which contains for some pages the infobox, the different sections identified by '==History==', and references (<ref>).

From the title page, we can access the corresponding Wikipedia page by adding underscores instead of spaces. For example, to access 'Colegio de Santa Cruz de Tlatelolco' page, the URL is `https://en.wikipedia.org/wiki/Colegio_de_Santa_Cruz_de_Tlatelolco`.

The XML file has 171742 Wikipedia pages. Only 5 pages have an empty content from the <text> tag. They will therefore not be used to build the recommender system.

# 3 Concurrence study

## 3.1 Approaches

The following section presents the current approaches that exist to implement a recommender system on content-based filtering.

**Doc2Vec** [10] is a method to obtain content-based representations of document data to a vector space model. The algorithm is based on word2vec which represents word in a vector space model, and adopted to sentences, paragraphs, and documents. It uses neural network to save the context and the semantic words unlike traditional bag-of-words approaches which eliminate the context.

**K-Nearest Neighbors (KNN)** is a supervised classification algorithm which is based exclusively on the choice of the classification metric. It is a non-parametric machine learning method which assumes that similar things are closed to each other. The idea behind KNN is that, from a labeled database, we can estimate the class of a new data item by looking at what is the majority class of the k closest neighboring data (hence the name of the algorithm). The only parameter to set is k, the number of neighbors to consider. In the work of [1], they proved that a Real-Time recommendation engine powered by KNN classification model implemented with Euclidean distance method is capable of producing useful, quite good and accurate classifications. Also, competent of recommendations to the client at any time based on his immediate requirement rather than information based on his previous visit to the site.

**TF.IDF** short for **Term Frequency-Inverse Document Frequency**, is a statistical measure intended to evaluate how a word is relevant to a document among a collection of documents. A study in 2016 [2] showed that 83% of text-based recommender systems in digital libraries use tf-idf. It also has shown its efficiency in information retrieval and text mining, where variations of the tf-idf scheme are used by search engines as an essential tool in scoring and ranking a document's relevance given a use query. It can be calculated by multiplying two metrics: the term frequency and the inverse document frequency.

- **Term Frequency** (TF) measures the frequency of occurrence of the terms in the document.
- **Inverse Document Frequency** (IDF) measures how important a term is. While computing tf, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones.

**Count-vectorizer** is provided by scikit-learn library and converts a collection of text to a matrix of token counts.

**Graph-Convolution-based approach**, its acronym of GCN, is an approach for semi-supervised learning on graph-structured data. Handling directly on graphs, the GCN approach is relying on an efficient variant of Convolutional Neural Networks (CNN), and the alternative of the architecture of CNN is stimulated through localized first order approximation of spectral graph convolutions. Moreover, in the number of graph edges, its model scales linearly and learns hidden layer representations that encode both features of nodes and local graph structure [9].

**Latent Dirichlet Allocation (LDA)** is a generative statistical model which explains observation groups by unobserved ones. This is the reason why certain part of the data are similar.

It has already been used to generate Wikipedia links [3].

**Bag of Words** (BoW) is a method to represent text data with the algorithm of machine learning. It is easy to understand and implement. Moreover, it is known for its success in talking about the problem in language modeling and document classification. The method represents text which describes the occurrence of words in the documents which relates to a known word in the vocabulary and the presence of known words in the measurement.

**The similarity matrix** is to represent the distance between two sets of data, and its results of the distances are mapped to the value to create the representation.

In [13], they presented a **deep learning architecture with Recurrent Neural Networks** in a top-N content-based recommendation scenario. In more details, they proposed a deep architecture based on Long Short Term Memory (LSTM) networks to jointly learn two embeddings, the first one representing the items to be recommended, and the second one to encode the preferences of the user. From these two embeddings, they calculated the relevance score of each item for a specific user thanks to a logistic regression layer and were able to return the top-N items as recommendations. By comparing there work with two baselines based on neural networks, Word2Vec and Doc2Vec, and state-of-the-art algorithms for collaborative filtering, they showed the effectiveness of their approach. But such systems have some drawbacks. It requires a lot of data, extensive hyper-parameter tuning and the interpretability and explainability can be difficult.

### 3.2 Common Evaluation Metrics

Such systems require a proper evaluation. Many evaluation metrics are available for recommender systems and each has its own pros and cons. We present here a short summary of some of them.

**Precision and Recall** are binary metrics used to evaluate models with binary output. Table 1 present the results from a user that will be used to compute the precision and recall scores [7]. Precision is the ratio between the True Positives and all the Positives (TP / (TP+FP)).

| | Recommended | Not recommended |
|---|---|---|
| Preferred | True-Positive (TP) | False-Negative (FN) |
| Not preferred | False-Positive (FP) | True-Negative (TN) |

Table 1: Classification of the possible results of a recommendation of an item to a user.

Recall corresponds to the ratio between True Positives and True Positives + False Negatives (TP / (TP+FP)).

**Mean Average Precision (MAP)** computes the mean of the Average Precision (AP) over all the users. It takes in a ranked list of the N recommendations and compares it to a list of the true set of "correct" or "relevant" recommendations for that user.

**Area Under The Curve - Receiver Operating Characteristics Curve** AUC - ROC is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how

much the model is capable of distinguishing between classes.

**Normalized Discounted Cumulative Gain (NDCG)** is the metric of measuring ranking quality. It is mostly used in information retrieval problems such as measuring the effectiveness of the search engine algorithm by ranking the articles it displays according to their relevance in terms of the search keyword.

**Mean Absolute Error (MAE)** measures the errors between observed pairs expressing the same phenomenon.

**Root Mean Square Error (RMSE)** measures differences between values predicted (by a model) and the values observed.

**Novelty and Diversity** are the key dimensions of recommendation utility, and a fundamental research direction to keep making progress in the field.

Content-based filtering is useful when they do not exist any suggestions; and do not depend on the users. Moreover, the recommendations are all based on known features. Nevertheless, it is limited on the information available in the content and does not allow suggestions on new things.

# 4 Recommender system strategy

## 4.1 The main strategies of recommender systems

### 4.1.1 Collaborative Filtering (CF)

CF needs the user historical preferences on a set of items assuming that the users who have agreed in the past tend to also agree in the future, in this approach users are modeled as a simple list containing the ratings provided by the user for some item [6]. It can be expressed by two categories. **Explicit Rating** is a rate given by the user to an item, like 5 stars to a film in Netflix. **Implicit Rating** reflexes indirectly the user preferences, for example, clicks numbers, pages views, time spent on a website, etc.

### 4.1.2 Content-Based Filtering (CBF)

CBF algorithms consider the history of the individual users' preferences and try to learn a preference model which is based on a feature-based representation of the content of recommendable items [11]. Additionally, this approach recommends any similar items that they are based on specific notation of the domain or the content of the item. This approach also enlarges to cases where there is availability of the metadata, and the information contain extra knowledge in recommendable items including features, category assignment, metabase and tags which provided by users & comments, and other related textual contents.

### 4.1.3 Demographic Filtering (DF)

Demographic Filtering classifies the user based on personal attributes and makes recommendations based on demographic classes. Demographic classes are divided following user's personal attributes, these classes are the input data. The objective of this process is to find the classes of people who like a certain product, the customers provide the personal data via surveys that they fill in during the registration process or can be extracted from the purchasing history of the users. This technique may not require collecting the complex data such as history of users' purchases and ratings [8].

### 4.1.4 Knowledge-Based Filtering (KBF)

A recommender system is knowledge-based when the recommendations are based on the knowledge of the users and items [5]. This system retrieves what items meets the users' requirements in order to propose similar ones [14]. Some of well-known examples are e-commerce websites. For example, if the user wants to buy a clothe, he/she can based his/her research on the type, the price, the color, the size, etc. The recommendation algorithm works as follows:

1. The user makes a query in the system.
2. Relevant rules are found, linked to the domain knowledge. The rules are of the form "if-this-then-that".
3. A single database query is constructed by combining the requirements. This final query will be used by the recommender system to retrieve relevant items from the database.

### 4.1.5 Hybrid approach

The main idea here is to combine some of existing strategies [6], explained above. A famous example of such system is Netflix [12] which combines collaborative filtering (cf. section 4.1.1) and content-based filtering (cf. section 4.1.2). The system compares watching and searching

habits of similar users, but it is also capable to make movies suggestions about one or more that share characteristics with films that a user has rated highly.

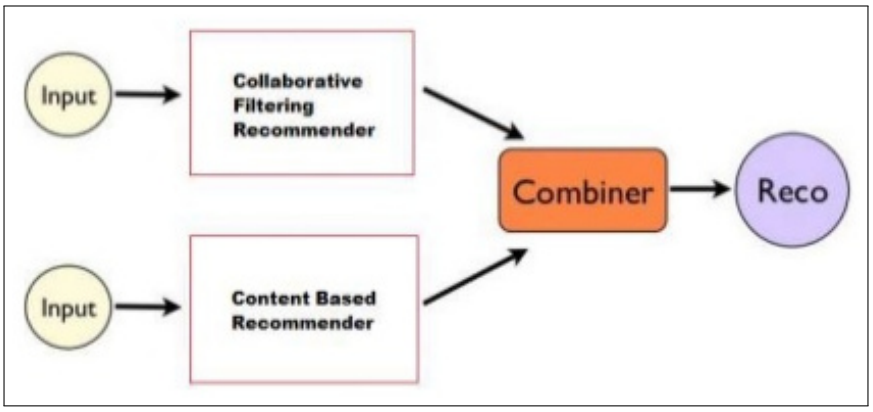


Figure 2: Example of a hybrid recommender system.

In general, hybrid techniques use collaborative filtering and another approach. Some of hybridization methods [4] are listed bellow:

- Weighted,
- Switching,
- Mixed,
- Feature combination,
- Cascade,
- Feature augmentation,
- Meta-level,
- EntreeC.

### 4.2 Our proposal

Our proposal will use content-based filtering. From a query of a user, which corresponds to a Wikipedia link, our system will use the content of the page to recommend 10 other links. By content of the page, we mean the title and the text that contains the page. Some preprocessing steps to remove the punctuation, the stop words such as "and", "but", etc. will be implemented.

### 4.3 Libraries

We used PYTHON and the main libraries for this projects are GENSIM & SCIKIT-LEARN.

### 4.4 Chosen approaches

Two approaches will be used to generate the links. The first one is Doc2Vec and the second one is TF-IDF (cf. section 3). In the beginning, we were thinking of using KNN but, because it is a supervised approach, it did not correspond to the type of data we had.

# 5 Implementation of the chosen approaches

This section explains the work we have done to implement both approaches. We will also provide an evaluation for both and discuss the challenges.

## 5.1 User interface

Before applying the different approaches to generate 10 links, we had to retrieve the request of the user, i.e. the Wikipedia link, and check if it exists in our database. To do so, we constructed a user interface which first retrieves the request of the user and then checks if the following requirements are observed:

- if it is in the form of `https://en.wikipedia.org/wiki/` + name of the page,
- if the name of the page is in the database (indeed, we can't generate the related links if we don't have it in our knowledge base).

Figure 3 bellow shows the case when the request follows the corresponding requirements.

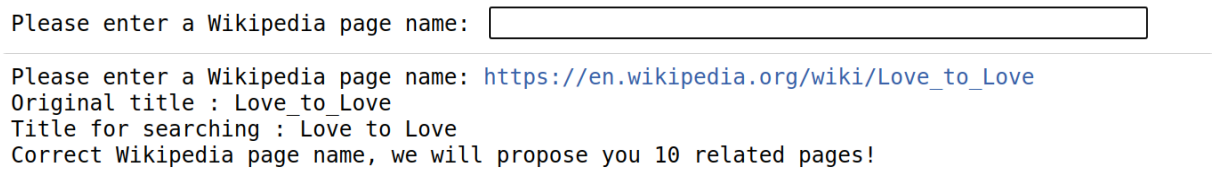


Figure 3: User interface when the request is correct.

If the user request is incorrect (for example with the link `https://en.wikipedia.org/wiki/Impractical_Jokers`), we suggest some existing pages from our database related to the words the user wrote in the corresponding form. Figure 4 shows the all procedure.


Figure 4: User interface when the request is incorrect.

Until the user doesn't fill a correct Wikipedia link, the interface requests another link. The user has also the choice to quit the interface with the keyword *exit*. Finally, the interface recommends different links depending on the chosen approach.

## 5.2 Methods

### 5.2.1 Doc2Vec

As explained in section 3.1, doc2vec is based on word2vec and the main motivation is to represent documents into numeric values. It was presented in 2014 by Mikilov and Le [10]. The main difference between word2vec and doc2vec is that doc2vec solves the problem of missed logical structure in documents by adding another vector (Paragraph ID). There are two flavors:

- Distributed Memory Model of Paragraph Vectors (PV-DM), it is is similar to Continuous-Bag-of-Words (CBOW) model in word2vec which attempts to guess the output (target word) from its neighboring words (context words) with the addition of the paragraph ID (i.e. Figure 5).
- Distributed Bag of Words version of Paragraph Vector (PV-DBOW), it is similar to skip-gram model of word2vec which guesses the context words from a target word (i.e. Figure 6).

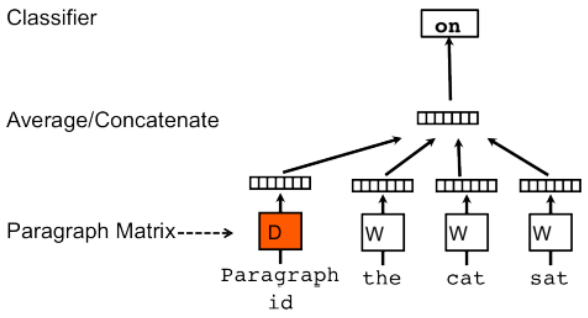


Figure 5: PV-DM model

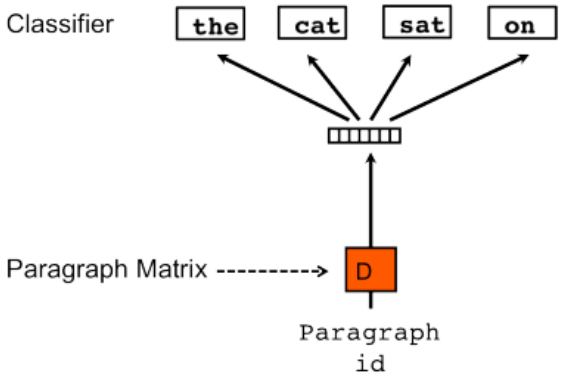


Figure 6: PV-DBOW model

We trained both models on the entire dataset chosen previously, as explained in section 2. Doc2Vec uses the complete dataset as input to be trained. The first model (PV-DBOW) took more than 1 hour to be loaded and the second one (PV-DM) took around 45 minutes. However, for the evaluation we choose to only focus on the first model.

```
cores = multiprocessing.cpu_count()

models = [
    # PV-DBOW
    Doc2Vec(dm=0, dbow_words=1, vector_size=200, window=8, min_count=19, epochs=10, workers=cores),
    # PV-DM w/average
    Doc2Vec(dm=1, dm_mean=1, vector_size=200, window=8, min_count=19, epochs=10, workers=cores),
]
```

Figure 7: Parameters used to train Doc2Vec models.

Onces the models are ready we can ask for the most similar pages from an input query (in textual format). The model ouput the name of the page and the score of the vectors, as shown in Figure 8.

```
['love', 'to', 'love']
[('Love to Love', 0.7007774114608765),
 ('The Romance of Kenny G', 0.6808857917785645),
 ('Just Like Heaven', 0.6676369905471802),
 ('Comfort and Joy', 0.6651524901390076),
 ('Sour (album)', 0.6620450019836426),
 ('Talking in Your Sleep', 0.6607742309570312),
 ('The Very Best of Kenny G', 0.6546471118927002),
 ('The Collection (Kenny G album)', 0.6523045301437378),
 ('Meet You There (album)', 0.6512295603752136),
 ('Soldier (Neil Young song)', 0.6511427760124207)]
```

Figure 8: Output of Doc2Vec model.

Figure 9 shows an example of the results we could get from the Wikipedia page `https://en.wikipedia.org/wiki/Love_to_Love`.

```
Please enter a Wikipedia page name: https://en.wikipedia.org/wiki/Love_to_Love
Original title : Love_to_Love
Title for searching : Love to Love
Correct Wikipedia page name, we will propose you 10 related pages!
======Model======
https://en.wikipedia.org/wiki/The_Romance_of_Kenny_G
https://en.wikipedia.org/wiki/Love_to_Love
https://en.wikipedia.org/wiki/The_Collection_(Kenny_G_album)
https://en.wikipedia.org/wiki/Will_You_Still_Love_Me?_(EP)
https://en.wikipedia.org/wiki/Lost_and_Gone
https://en.wikipedia.org/wiki/Soldier_(Neil_Young_song)
https://en.wikipedia.org/wiki/The_Very_Best_of_Kenny_G
https://en.wikipedia.org/wiki/Sour_(album)
https://en.wikipedia.org/wiki/Just_Like_Heaven
https://en.wikipedia.org/wiki/Stop_the_Machine
======Model======
https://en.wikipedia.org/wiki/Speed_skating_at_the_1999_Asian_Winter_Games
https://en.wikipedia.org/wiki/Frankl
https://en.wikipedia.org/wiki/Bible_Christian_Mission
https://en.wikipedia.org/wiki/Hellstrom
https://en.wikipedia.org/wiki/List_of_foliage_plant_diseases_(Bromeliaceae)
https://en.wikipedia.org/wiki/Mihajlovo
https://en.wikipedia.org/wiki/Durand_Township,_Winnebago_County,_Illinois
https://en.wikipedia.org/wiki/Diving_at_the_1920_Summer_Olympics_–_Men's_plain_high_diving
https://en.wikipedia.org/wiki/Chetin_Kazak
https://en.wikipedia.org/wiki/Diving_at_the_1972_Summer_Olympics_–_Men's_3_metre_springboard
```

Figure 9: Results from the two Doc2Vec models of the Wikipedia page name *Love to Love*.

### 5.2.2 TF-IDF

As explained in section 3.1, TF-IDF is a method which calculates a word's frequency in a collection of documents. It evaluates the importance of a word in these documents. The algorithm works as follows:

1. Computation of the TF - the frequency of a word in the document, TF(t,d) = frequency of t in d / maximal frequency of a term in d
2. Measurement of the IDF - how often term appears in all documents, IDF(t) = log(N/$n_t$) where $N$ is the total number of documents and $n_t$ refers to the number of documents containing $t$.

The final TFIDF matrix is the dot product between TF and IDF.

In order to run the TF-IDF algorithm, several steps were implemented:

- Extraction of the data from XML file and store them in a dataframe (i.e. section 5.2.2.1).
- Preprocessing of the data (i.e. section 5.2.2.2).
- Compute the TF-IDF matrix (i.e. section 5.2.2.3).
- Calculate the cosine similarity from the resulting matrix (i.e. section 5.2.2.4).
- Recommend 10 links from the user request based on the cosine similarity scores (i.e. section 5.2.2.5).

#### 5.2.2.1 Extraction of the XML data

As we explained in section 2, we took a sample of data from Wikipedia database. To extract the content of the data that are in an XML file, we used the XML.ETREE.ELEMENTTREE API, a simple and efficient API for parsing and creating XML data. We decided to keep only certain information from it that are described in the following:

- the title of the Wikipedia page, stored in a <title> tag,
- the id of the Wikipedia page, stored in an <id> tag,
- the content of the page, stored in a <text> tag.

The selected information were finally stored in a pandas dataframe (i.e. figure 10). We decided to delete some Wikipedia pages due to lack of content information (pages with a title of the form *Portal:****, *File:****, *Category:****, *JPG:****, *PNG:****, *jpg:****, *Wikipedia:****, *Template:****). It reduced the data from 171742 elements to 133224 elements.

| | Title | ID | Text |
|---|---|---|---|
| 0 | Chestnut Ridge Middle School | 8554860 | #REDIRECT[[Washington Township Public School D... |
| 1 | Colegio de Santa Cruz de Tlatelolco | 8554864 | {{Infobox university\n\|name = Col... |
| 2 | Impractical joker (garfield) | 8554867 | #REDIRECT [[List of Garfield and Friends episo... |
| 3 | National Council of Teachers | 8554873 | '''National Council of Teachers''' may refer t... |
| 4 | Shuo Wang | 8554878 | #REDIRECT [[Wang Shuo]] |
| 5 | The impractical joker garfield and friends | 8554883 | #REDIRECT [[List of Garfield and Friends episo... |
| 6 | Order of battle at Beiping–Tianjin | 8554884 | '''Peiking Tientsin Operation''' (July–August ... |
| 7 | Gulshani | 8554885 | {{about\|the Sufi order\|the demonym of Gulshan\|... |
| 8 | The impractical joker garfield & friends | 8554892 | #REDIRECT [[List of Garfield and Friends episo... |
| 9 | The impractical joker garfield | 8554898 | #REDIRECT [[List of Garfield and Friends episo... |

Figure 10: Beginning of the dataframe with title, id and text information for each page.

#### 5.2.2.2 Preprocessing

We preprocessed the data in order to reduce its size and get rid of the words that were not semantically important e.g. stop-words, numbers, etc. Using NLTK and REGEX, we wrote functions to do the following:

- Remove HTML tags like: <ref>, <img>.
- Remove URLs: we remove the links using the regular expressions.
- Remove punctuation marks: also using regular expressions.
- Remove stop words: we remove stop-words ("the", "is", ...) using NLTK stop-words list.
- Remove numerical data.

#### 5.2.2.3 TF-IDF matrix

The TF-IDF matrix was computed thanks to the command lines

```
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df = 0, max_features
= 1000, stop_words = 'english')
tfidf_matrix = tf.fit_transform(dataframe['Text'])
pickle.dump(tfidf_matrix, open("tfidf.pkl", "wb"))
```

From TFIDFVectorizer, **analyzer** refers to the type of feature, here words, **ngram_range** refers to the range of n-values for different n-grams to be extracted, **min_df** is to ignore terms in the vocabulary that have a document frequency strictly lower than the given threshold, **max_features** to build a vocabulary that only consider the top max_features ordered by term frequency across the corpus and **stop_words**. Then, we fit and transformed the textual data before feeding the algorithm.

Before computing the TF-IDF matrix on the content of Wikipedia pages, we made a first test with only titles. It is also important to mention that, for this test, we only used a subset of our data to help us understand the method before running the code on the whole dataset.

```
Chestnut Ridge Middle School
{'Chestnut': 0.25, 'Ridge': 0.25, 'Middle': 0.25, 'School': 0.25, 'Colegio': 0, 'de': 0, 'Santa': 0, 'Cruz': 0, 'Tlate
lolco': 0, 'Template:US-gov-bio-stub': 0, 'Impractical': 0, 'joker': 0, '(garfield)': 0, 'File:The': 0, 'Imperial': 0,
'Dowager': 0, 'Empress': 0, 'Yehenara.PNG': 0, 'National': 0, 'Council': 0, 'of': 0, 'Teachers': 0, 'Shuo': 0, 'Wang':
0, 'The': 0, 'impractical': 0, 'garfield': 0, 'and': 0, 'friends': 0, 'Order': 0, 'battle': 0, 'at': 0, 'Beiping–Tianj
in': 0, 'Gulshani': 0}
----------
Colegio de Santa Cruz de Tlatelolco
{'Chestnut': 0, 'Ridge': 0, 'Middle': 0, 'School': 0, 'Colegio': 0.16666666666666666, 'de': 0.2329900014453396, 'Santa
': 0.16666666666666666, 'Cruz': 0.16666666666666666, 'Tlatelolco': 0.16666666666666666, 'Template:US-gov-bio-stub': 0,
'Impractical': 0, 'joker': 0, '(garfield)': 0, 'File:The': 0, 'Imperial': 0, 'Dowager': 0, 'Empress': 0, 'Yehenara.PNG
': 0, 'National': 0, 'Council': 0, 'of': 0, 'Teachers': 0, 'Shuo': 0, 'Wang': 0, 'The': 0, 'impractical': 0, 'garfield
': 0, 'and': 0, 'friends': 0, 'Order': 0, 'battle': 0, 'at': 0, 'Beiping–Tianjin': 0, 'Gulshani': 0}
```

Figure 11: First TF-IDF results from the titles.

From Figure 11, we can see that the results only from titles are not interesting because the importance of a word is estimated only in the title level.

Therefore, we ran the algorithm in the content of Wikipedia pages, corresponding to the data of approximately size 130000. Because our personal computers were computationally limited, we decided to use Grid5000, a large-scale and flexible testbed for experiment-driven research.

It allowed us to book GPUs to improve the calculation capacity needed. The running process took approximately 2 hours and the resulting matrix was save in a binary file with PICKLE library. The binary file is of size 60 Mo. When running the TF-IDF method, we tested different ngram_range and max_features values but the larger the values, the larger the file size and the longer the calculation time.

#### 5.2.2.4 Cosine similarities

The cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space. This measure is obtained thanks to the given formula:

$$similarity(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| * \|\vec{b}\|}$$

It corresponds to the dot product of the two vectors divided by the product of the two vectors' lengths (or magnitudes).

The cosine similarity was computed thanks to the command line:

```
cosine_similarities = cosine_similarity(tfidf, tfidf)
```

When running the following line, the issue was a memory one. The solution to get cosine similarity scores was to reduce the size of the matrix with:

```
cosine_similarities = cosine_similarity(tfidf, tfidf[:50000])
```

Figure 14 shows the resulting array where each row and column correspond to one Wikipedia Page. The highest the cosine similarity score, the better the similarity is.

```
[[1.         0.08876145 0.         ... 0.         0.         0.        ]
 [0.08876145 1.         0.00862954 ... 0.09056698 0.         0.        ]
 [0.         0.00862954 1.         ... 0.02604671 0.05420052 0.16458987]
 ...
 [0.         0.02352993 0.         ... 0.06226215 0.         0.        ]
 [0.         0.01318819 0.01133071 ... 0.00997876 0.         0.        ]
 [0.01710205 0.03296025 0.0090965  ... 0.00222621 0.         0.        ]]
```

Figure 12: Cosine similarity matrix.

#### 5.2.2.5 Recommendation links

Figure 13 shows an example of the results we could get from the Wikipedia page `https://en.wikipedia.org/wiki/Love_to_Love`. As we can notice, each link has a similarity score.

```
Request :  https://en.wikipedia.org/wiki/Love_to_Love
Recommending 10 links similar to Love to Love page...

1. https://en.wikipedia.org/wiki/Pjetër_Dungu (score:0.083)

2. https://en.wikipedia.org/wiki/New_York_State_Route_52_Business (score:0.061)

3. https://en.wikipedia.org/wiki/Good_Night,_Little_Ones! (score:0.046)

4. https://en.wikipedia.org/wiki/The_Silence_of_the_Lambs (score:0.045)

5. https://en.wikipedia.org/wiki/Criminal_court (score:0.024)

6. https://en.wikipedia.org/wiki/1982_Topps (score:0.009)

7. https://en.wikipedia.org/wiki/Gene_D._Block (score:0.008)

8. https://en.wikipedia.org/wiki/Judicial_intern (score:0.008)

9. https://en.wikipedia.org/wiki/Colegio_de_Santa_Cruz_de_Tlatelolco (score:0.007)

10. https://en.wikipedia.org/wiki/Diarmuid_O'Neill (score:0.006)
```

Figure 13: Recommendation links for the Wikipedia page name *Love to Love* and their associated cosine similarity score.

# 6 Analysis

Having successfully experimented the two models, it is time for the evaluation of the results, we validated the results by conducting a subjective evaluation which had been made through Google Survey Form.

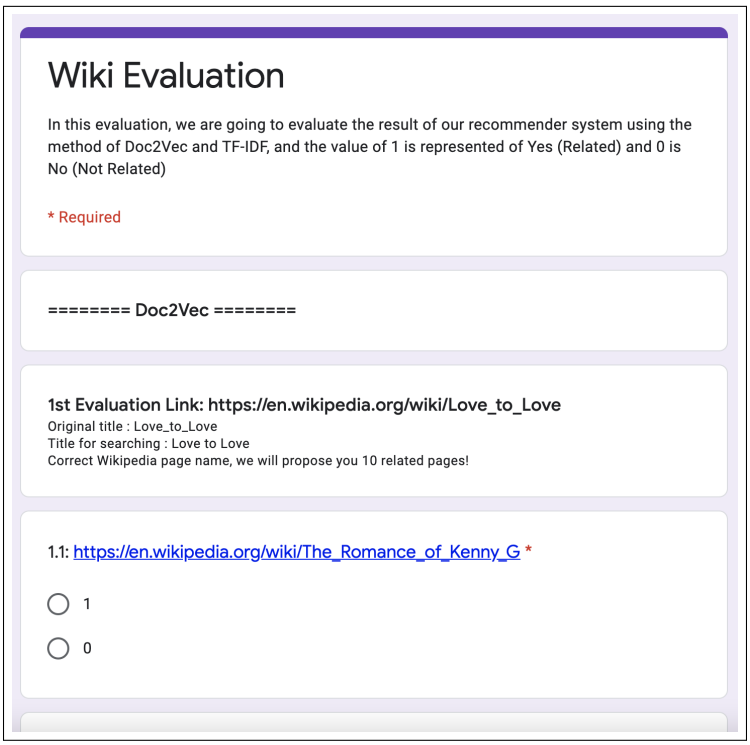


Figure 14: Evaluation on Google Survey Form.

### 6.1 Evaluation

Like we stated above, to estimate the performance of our models, we conducted a subjective evaluation which is evaluated by our team consisting of 5 people. The evaluation is completed by putting all of the generated results from the models and filled into Google Survey Form in order to validate if the generated results are relevant to the link which the users input. Moreover, each of the related link was chosen as the value of 1 which means as "relevant", and the value of 0 is the notation of "not relevant". In each model, there are 10 suggested links to pick one of the two options that provides the best correlation. Upon the completion, we calculated the results and produced the value of average and the standard deviation for each model.

| **Doc2Vec** | **Total Score** | **Average Score** |
|---|---|---|
| 1st Evaluation Link | **31** | **62%** |
| 2nd Evaluation Link | 26 | 52% |
| 3rd Evaluation Link | 2 | 4% |

Table 2: Doc2Vec evaluation results.

| **TF-IDF** | **Total Score** | **Average Score** |
|---|---|---|
| 1st Evaluation Link | 5 | 10% |
| 2nd Evaluation Link | **40** | **80%** |
| 3rd Evaluation Link | 14 | 28% |

Table 3: TF-IDF evaluation results.

As the result, in Table 2 and Table 3, we can see both of the result from these two models, and the highest average score that we obtained from Doc2Vec is 31 over the 50 score which corresponds to 62%. TF-IDF gave a score of 40 over 50 which is the highest on average score giving 80%. However, in Table 4, the result for the two models had the same value meanwhile the TF-IDF got better score on standard deviation at 36%.

| **Methods** | **Avg Score** | **Standard Deviation** |
|---|---|---|
| **Doc2Vec** | 39% | **31%** |
| **TF-IDF** | 39% | **36%** |

Table 4: TF-IDF & Doc2Vec results for avg and standard deviation.

### 6.2 Challenges

From the results, we were not able to decide which model was computing the best results. Indeed, the evaluation was limited to our group of 5 people and on 3 Wikipedia pages for each approach. Because we only used a small amount of data, we could imagine to have this result, meaning that some Wikipedia Pages did not have correlated pages in our database.

From the two approaches we implemented, several disadvantages are seen. TF-IDF is based on the BoW model, therefore it does not capture position in text, semantics, co-occurrences in different documents, etc. In other words, this approach is only useful in a lexical level. It can not capture the semantic information (topic modeling) which could have been useful in this

project. The highest TF-IDF words of a document may not make sense with the topic of the document.

TD-IDF generates a huge matrix. Computing the cosine similarities between two huge matrices caused a memory error, we had to reduce the size of the second matrix when calculating the similarity. However, this approach is not efficient and leads to a huge loss of data and inaccurate results. We investigated on this issue and we concluded that it is possible to calculate the similarities in parallel since there are no dependencies in these calculations. Parallelism can be done using `joblib`, a Python library that can simply turn a Python code into parallel computing mode, and of course, increase the computing speed. Unfortunately, due to the time constrains we could not apply this mechanism.

Doc2Vec takes times to load the models on the chosen data. But once they are ready, the results are rapidly generated. As said previously, we only evaluated the first model (PV-DBOW) because we found out that the results were more correlated.

# 7 Conclusion

This work goal was to understand and implement a methodology in order to recommend links based on Wikipedia data. We focused our project on the content-based filtering methodology and decided to implement two approaches: Doc2Vec and TFIDF. The given approaches were challenging, we faced few issues when running the algorithms. However, we were able to recommend links for both and make a subjective evaluation. Future work would involve improving the algorithms by increasing the computational capacity and test different parameters to get more tune-fining results. We also only implemented two approaches. Some interesting works could be done using deep learning techniques.

The organisation of the work was splitted in the following way:

| **Tasks** | **Done by** |
|---|---|
| Extraction of the data | Cécile |
| User interface | Cécile |
| Doc2Vec approach | Ludivine |
| Preprocessing | Fatima |
| TFIDF approach | Fatima & Asmaa |
| Subjective evaluation form | Chanoudom |

# References


[1] David Adedayo Adeniyi, Zhaoqiang Wei, and Y Yongquan. "Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method". In: *Applied Computing and Informatics* 12.1 (2016), pp. 90–108.

[2] Joeran Beel et al. "Research-paper recommender systems : a literature survey". In: *International Journal on Digital Libraries* 17.4 (2016), pp. 305–338. ISSN: 1432-5012. DOI: 10.1007/s00799-015-0156-0.

[3] Karan Bhanot. *Building an Article Recommender using LDA*. https://towardsdatascience.com/lets-build-an-article-recommender-using-lda-f22d71b7143e. Online; accessed 31 January 2021. 2019.

[4] Robin Burke. "Hybrid recommender systems: Survey and experiments". In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[5] Robin Burke. "Knowledge-based recommender systems". In: *Encyclopedia of library and information systems* 69.Supplement 32 (2000), pp. 175–186.

[6] Erion Çano and Maurizio Morisio. "Hybrid Recommender Systems: A Systematic Literature Review". In: *CoRR* abs/1901.03888 (2019). arXiv: 1901.03888. URL: http://arxiv.org/abs/1901.03888.

[7] Asela Gunawardana and Guy Shani. "A survey of accuracy evaluation metrics of recommendation tasks." In: *Journal of Machine Learning Research* 10.12 (2009).

[8] L. Chameikho Iateilang Ryngksai. "Recommender Systems: Types of Filtering Techniques". In: *International Journal of Engineering Research  Technology* 13 (November-2014).

[9] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv preprint arXiv:1609.02907* (2017).

[10] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. PMLR. 2014, pp. 1188–1196.

[11] Pasquale Lops et al. "Trends in content-based recommendation". In: *User Modeling and User-Adapted Interaction* 29.2 (2019), pp. 239–249.

[12] Saimadhu Polamuri. *Introduction to recommendation engine*. Online; accessed 1 February 2021. 2015.

[13] Alessandro Suglia et al. "A Deep Architecture for Content-Based Recommendations Exploiting Recurrent Neural Networks". In: *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. UMAP '17. Bratislava, Slovakia: Association for Computing Machinery, 2017, pp. 202–211. ISBN: 9781450346351. DOI: 10.1145/3079628.3079684. URL: https://doi.org/10.1145/3079628.3079684.

[14] Jackson Wu. *Knowledge-Based Recommender Systems: An Overview*. https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba. Online; accessed 1 February 2021. 2019.