

**Recommender System for TV & Movies micro-genres**

**Case 3: Think Analytics**

**MS984 Data Analytics in Practice**

**Group 4**

Mr Sourabh Shubhanandan Mahajan

Mr Yash Sharma

Mr Antonio Marchi

Mr Sven Kruthoff

Mr Ruixian Zhao

Date: 23th Nov 2018

# Executive Summary

This report, commissioned by ThinkAnalytics Ltd. as a piece of work for a client of theirs, describes a conceptual model for a recommender system based on Probabilistic Latent Semantic Indexing (PLSI). In the report, a solution to retrieving documents based on sub-genres is suggested by fitting an EM algorithm to PLSI concepts to improve the functionality and precision of recommendations. We test our conceptual model against empirical evidence of similar industrial applications to validate its robustness and conceptual validity and discuss positive findings.

# Table of Contents

[Executive Summary 0](#_Toc531046745)

[Table of Contents 0](#_Toc531046746)

[Introduction to the Conceptual Model - 1 -](#_Toc531046747)

[Conceptual Model Description - 2 -](#_Toc531046748)

[Metadata Gathering and Pre-Processing - 2 -](#_Toc531046749)

[PLSI Processing of Metadata - 3 -](#_Toc531046750)

[Handling Queries and Making Recommendations - 3 -](#_Toc531046751)

[Evaluation & Discussion - 4 -](#_Toc531046752)

[Conclusion - 4 -](#_Toc531046753)

[References - 5 -](#_Toc531046754)

[Appendix - 6 -](#_Toc531046755)

# Introduction to the Conceptual Model

As per client requirements, the model needs to fulfil 3 main requirements. It needs to make sense of a descriptive metadata database, by grouping documents into categories (i.e. subgenres), and use those categories as the backbone of a recommender system able to recognise movies by subgenre. Secondly, the recommender system should be able to handle queries of users and retrieve appropriate documents with adequate precision. Our conceptual system is made of a sequence of main processes which at each stage source, cleanse, pre-process metadata and store them in MongoDB databases. When a query reaches the system, our conceptual model, powered by an Expectation Maximisation algorithm, decomposes it and retrieves the documents which have the highest probability of belonging to the categories the query refers to. Categories the queries belong to are inferred through Probabilistic Latent Semantic Indexing (PLSI) and are matched to queries through the EM algorithm. This helps to not only retrieve movies that are exact matches to keywords within the queries, but also keywords describing the movies associated with the keywords in the query. Finally, the system should be able to not only match movies to queries, that have the same keywords, but also recommend related movies that match certain categories within the queries based on popularity or similarity.



*Figure 1*. Visualisation of the Conceptual Model

# Conceptual Model Description

## Metadata Gathering and Pre-Processing

Metadata refers to a pool of raw information about a movie. It includes elements such as: title, synopsis and keywords. A representation of a sample document's related metadata is illustrated in Appendix 1.

A dynamic metadata database, where metadata on various movies is collected and stored is an integral feature of our conceptual model. The database can be built with MongoDB and programmed by a python script. A cronjob should be written using python for the system to automatically, at 1 am every day, pull metadata from three public domain sources, respectively: Wikipedia, IMDB and OMDB. This is done by sending a request to their dedicated API system for any document published within the previous 24 hours (1)(2). Scrapping data from multiple sources serves the purpose of building a comprehensive database of synopsis metadata, which will form the backbone of the model. MongoDB is the appropriate type of database to store the scraped data as it is robust and can manage large datasets. It also provides an environment that has been proven to enhance performance of the EM algorithm, which is the algorithm that powers our prototype to process metadata (3). Once metadata for a movie is retrieved from the relevant webpages, another cronjob instructed with python's word\_tokenize built-in functions can cleanse the document's synopsis from stopwords, punctuation and all words are transformed to lowercase letters to reduce the volume of metadata stored. All other categorical variables of a document only need made lowercases.

*"Harry Potter is a series of fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends."*

Will be processed and output the following:

*“harry potter series fantasy novels written british author j k rowling novels chronicle lives young wizard harry potter."*

Once the metadata is scrapped from all three websites, duplicate words are deleted and stored for PLSI processing.

## PLSI Processing of Metadata

PLSI is a statistical approach to automated text document indexing, based on the analysis of data co-occurrence. Data co-occurrence stands for those instances where two words appear in a document either next to each other or in an order that might be idiomatically classified. By analysing co-occurrence of words, rather than their mere frequency in a document, it is possible to capture synonymity as well as words that refer to a topic (4). Such system architecture enables our prototype system to infer more reliably to possible categories a user's query may refer to (i.e. Sub-genres), while also empowering the retrieval process by looking for words adjacent to those in the query.

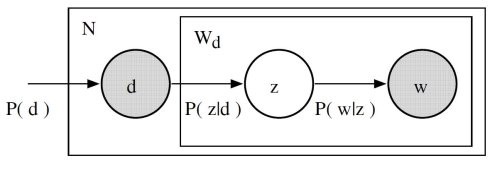
PLSI first performs Singular Vector Decomposition (SVD) to transform the words in a document (**w**) into singular vectors and compute their frequency distribution. It then converts co-occurrent words that indicate semantic proximity to a topic (**z**) into singular vectors, to again calculate frequency distributions within that document. The probability that a query should be matched to a specific document is illustrated in *Figure 2* as derived by the relationship *P*(**d**) = **Σ***P*(**z|d**)*P*(**w|z**) where *P*(**z|d**) refers to the probability that the document is related to a particular topic, while *P*(**w|z**) refers to the likelihood that the words in a query are related to a specific topic.

Figure 2: Relationship between *P*(**d**) and **Σ***P*(**z|d**)*P*(**w|z**)

## Handling Queries and Making Recommendations

Since it is assumed that each data point within the query [**w**] will belong to a latent category [**z**], the Expected Maximisation algorithm may be used to iteratively find the highest likelihood [***P***] that [**w**] in a query refer to topic (**z**), which in the case of our model refers to a sub-genre. The algorithm is fit for this purpose because of its soft clustering conception. Indeed, data objects may feature in more than one latent category (i.e. subgenre) but belong more to one category than another. Such 'membership' is recognised by the EM algorithm and is based on latent categories uncovered by PLSI. Once the documents that score highest are retrieved, they get ranked by their relevance of the primary keywords, meaning the exact match of query keywords and keywords describing the movies, and the secondary keywords, meaning those indirectly associated with the query keywords. To be able to further improve the recommendation system, the EM algorithm can further recommend movies that are associated with the answer to the initial query based on popularity, similarity of the movies and user behaviour. The recommendation based on popularity is derived from basic popularity of movies within the categories associated with the user query. The second method to recommend movies based on similarity of movies within each other is improving the query results by suggesting movies that for example share the same actors or directors, even though the movies are grouped in a different sub-genre. The third way to recommend movies within a query that are not directly connected to the initial query can be based on the user watching patterns. For example, if user 1 runs a query and this user likes movies A,B,C and D and user 2 ran similar queries and likes movies A,B,C and E, the movie E will be recommended within the query of user 1. The same process will be applied vice versa, so when user 2 runs a query movie D will be recommended to user 2. These additional methods based on the EM algorithm exist to improve the performance of the system for the user and further refine queries.

# Evaluation & Discussion

In evaluating any model, two parameters play an important role. The first one is the performance of the database and the following processing model. Secondly, the database should be stable and durable so that it can handle each request precisely and unbiasedly. Our choice to use a document type database MongoDB fulfils both parameters. This was empirically validated in practice by (Butler, 2018) who in his paper highlights that MongoDB is strong, fast and flexible when it comes to querying and data mining in connection with IMDB. Another parameter of evaluation is to check the capability of the processing model. The EM algorithm used widely to classify simple unstructured or structured text. Since the movie synopsis has plain text, EM has the capabilities to deal with that type of task (5).

# Conclusion

To summarize, the three given design requirements, dealing with categorising data, handling and improving queries are based on the basic working methods of probabilistic latent semantic indexing. Data is taken not only from movie related webpages but also Wikipedia, because it offers a rich synopsis. Data is transformed into a format that can be used for the two further steps of the system. In the second step, PLSI is used as a method to optimally match the user queries with not only the actual query keywords, but also keywords related to those in the query. Additionally, the system will improve queries by recommending other movies based on popularity, other peoples’ preferences and interactive content. In theory this system presents an optimal method of building and maintaining movie related queries and databases, but due to the lack of actual data, it cannot be certainly said that this method will work efficiently.

# 

# References

1. Readthedocs.org. (2018). *MediaWikiAPI | Read the Docs*. [online] Available at: <https://readthedocs.org/projects/mediawikiapi/> [Accessed 22 Nov. 2018].
2. Towards Data Science. (2018). *Step-by-step guide to build your own ‘mini IMDB’ database*. [online] Available at: <https://towardsdatascience.com/step-by-step-guide-to-build-your-own-mini-imdb-database-fc39af27d21b?_branch_match_id=576144636395779521> [Accessed 22 Nov. 2018].
3. Aarshay, J., *Quick Guide to Build a Recommendation System in Python*, Available at: <https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/> [Accessed 26 Nov. 2018].
4. Lavrenko, V. and Croft, W.B., 2017,. *Relevance-based language models*. In *ACM SIGIR  
   Forum* (Vol. 51, No. 2, pp. 260-267). ACM.
5. Nigam, K., *Text Classification from Labeled and Unlabeled Documents using EM*, 2018, [online] Link.springer.com. Available at: https://link.springer.com/content/pdf/10.1023%2FA%3A1007692713085.pdf [Accessed 26 Nov. 2018].

Additional Resources

Docs.mongodb.com. (2018). *Map-Reduce — MongoDB Manual*. [online] Available at: <https://docs.mongodb.com/manual/core/map-reduce/> [Accessed 26 Nov. 2018].

Ieeexplore.ieee.org. (2018). *Interface for querying and data mining for the IMDb dataset - IEEE Conference Publication*. [online] Available at: [https://ieeexplore.ieee.org/abstract/document/7494103/authors#authors](https://ieeexplore.ieee.org/abstract/document/7494103/authors) [Accessed 25 Nov. 2018].

Alghamdi, R. and Alfalfa, K., 2015. A survey of topic modelling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, *6*(1).

# Appendix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Title | Synopsis | Age Rating | Genre | Credits | Format | Release Year | Rating |
| Harry Potter [and the Sorcerer's Stone](https://www.imdb.com/title/tt0241527/?ref_=ttfc_fc_tt) | *Harry Potter is a series of fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry* | Children All Age | [Fantasy](https://en.wikipedia.org/wiki/Fantasy_literature), [drama](https://en.wikipedia.org/wiki/Drama),  [young adult fiction](https://en.wikipedia.org/wiki/Young_adult_fiction), [mystery](https://en.wikipedia.org/wiki/Mystery_(fiction)),  [thriller](https://en.wikipedia.org/wiki/Thriller_(genre)),  [Bildungsroman](https://en.wikipedia.org/wiki/Bildungsroman) | J.K. Rowling,  [Steve Kloves](https://www.imdb.com/name/nm0460141/?ref_=ttfc_fc_wr2) | Movie | 2001 | 7.8 |

*Appendix 1*: Sample of a metadata document