Text Feature Representation for Content Based Recommender Systems

Andrea Pagotto

December 21, 2017

1 Introduction

Humans use a lot of strategies in making decisions. Often in everyday life people rely on knowledge of their friends to give them advice on making a decision, such as choosing what place to eat, what movie to watch etc. However, this is a limited approach to gaining information about available products and services, since it relies on knowledge of only a few people. This is where automated recommender systems come into play. With the growing silos of data available, there is now much potential to harvest the knowledge of millions of users, and sources of data, to achieve good recommendations. There are many techniques in this field currently being used very successfully by large companies such as Amazon for product recommendations, Netflix for movie recommendations, etc. The most commonly used approach is called collaborative filtering, which relies on user ratings. This technique is applied very effectively, however there are issues with scalability and also with introducing new items into the system. With the wide spread growth of information, it would be good to consider incorporating additional information about the available products and services to see if this can improve results, and address these challenges.

The focus of this investigation will be to assess some methods of incorporating content information into generation of recommendations. The dataset used will be a standard recommender system testing dataset called "Movie Lens". Movies in this dataset have descriptions and keywords and other metadata that can be used to represent the content of the movie. This project will investigate natural language processing techniques applied to representing these movies, and assess the quality of the recommendations. This report will isolate this method of generating recommendations and assess it as a stand alone source of recommendations, as well as in a hybrid approach.

2 Motivations and Project Objectives

Word embeddings can capture information from numerous sources, for example word2vec by Google [12], and Glove word vectors [16] can be downloaded as pretrained vectors on all of wikipedia data. These word embeddings can capture the concept of a word in a low dimensional representation, with tunable dimension size. Everyday, more and more innovations are being done developing new word embeddings, included Facebook Fasttext, LexVec [17], Numberbatch [18], and many more. Also as the data sources available to train these embeddings continue to grow, these representations will get better and better. Therefore, the approach of using word embeddings to generate document (movie description) representations for the purpose of generating recommendations based on movie similarities will be investigated in this project. Other methods of representing the movie content will also be implemented and compared.

The objective of this project is to investigate vector representations of movie content, and assess the viability of using these representations in providing movie recommendations. This report will first explain the methods used, and past research in this area. Also, there is a file submitted (Jupyter Notebook) that will in detail show the implementation and experimentation done on the Movie Lens Dataset, and present results and observations. Finally, this report will conclude with some discussion of future directions of research.

3 Background and Literature Review

This section will first provide a general introduction to recommender system techniques, then provide some more specifics on past studies on content based recommender. Next, it will review methods in document and word representations, as well as measuring similarity between texts.

3.1 Recommender Systems

3.1.1 Classic Approaches

A recommender systems is a subclass of information filtering or information retrieval that seeks to predict how much a user in a set of users would like each item in a set of items. These systems, as mentioned are becoming increasingly popular in areas such as movies, music, books, etc. Recommenders produce recommendations through either collaborative or content based filtering.

Collaborative filtering builds a model from user's past behaviors, ie. by generating a matrix of items and users and filling in the rating each user gave to each item. The objective is to predict the unknown ratings for each user. This can be done with a variety of methods, many based on matrix factorization, to generate predictions of which items users would like strictly based on their previously submitted ratings, and the predictions of other users. An issue with this is known as the "cold start" problem. This means that significant informations is required about users and their predictions for each item before the system can begin making good predictions. Another issue is scalability which can occur in systems that have millions of users and products. Finally another problem is the sparsity of the matrix, i.e. the fact that the majority of the ratings matrix will be unknown [4].

3.1.2 Fully Content Based Approach

The idea behind content based recommendation systems is to create a profile for a user based on features of content that they liked.

Providing recommendations based fully on the content has been investigated in several past studies. The main study of reference in this area, is an investigation by Musto et al. [15], assessing word embedding techniques in content based recommendation systems. They applied several methods of representing content of the Movie Lens movies, using singular value decomposition, random indexing, and Google word2vec word embeddings on the movie descriptions. They found the results (precision, recall, F measure) for the recommendations they produced with these methods comparable to results from standard approaches of classic user-to-user recommendations and item-to-item recommendations. The best performance was done by the google word2vec description embedding method, however latent semantic indexing (singular value decomposition) was also a good approach and outperformed word2vec in some scenarios. The content based-recommendation framework used in this study will also be used in this project and will be explained in the implementation section.

Another study by Kim, S. et al [10] investigated generating movie similarities for recommendations based on movie scripts. The approach used in this study extracted features from documents represented as word count vectors. They evaluated results using a heuristic ranking comparison between the ranks of the movies similar to certain movies measured in different ways: cosine similarity, an on-line similarity score, and their new suggested method. They found their results in measuring document similarity showed improvements in some cases over the the cosine method. Their method of evaluating results will be used in this project, as a way to evaluate results.

Finally another recent study by Chen H., et al. [8] investigated movie feature representation for a neural network recommender. They applied embedding method (word2vec) to represent cast and crew as well as other meta-data features of the movies. These feature vectors were then used to train recommendations from a neural net. Document similarities were measure using cosine distance, and results were evaluated using a test and train set and then calculating the top k precision.

These studies inspired the current project objectives. They have showed that there is much opportunity in testing content based recommender systems, and applying natural language processing techniques to represent content. Therefore this project will investigate further some natural language processing techniques and perform some analysis and testing not yet investigated in current research. This project will test another set of word embeddings, the GloVe vectors, and also

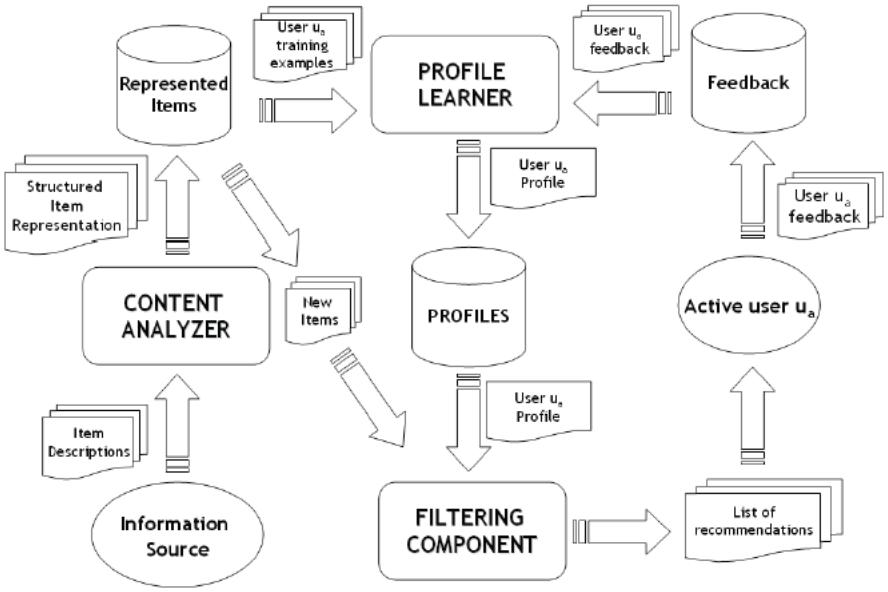

Figure 1: Hybrid Recommender Example Diagram

the basic term-frequency inverse document frequency representation of document descriptions and meta-data to perform content based recommendations.

3.1.3 Hybrid Approaches

One of the main motivations behind investigating content based recommender's in this study is because content recommender systems make up a part of larger hybrid recommendation schemes. It is worthwhile to test out the content based parts individually to assess the performance, before incorporating it as a module into a larger system. Larger system recommendation schemes, such as the example shown in the figure below, can become quite complex and therefore fully assessing the sub-modules can help to produce good performance [19].

3.2 Feature Representation

Feature representation of text is a widely studied area and there are many possible approaches to take to generate a numerical vector representations of text for use in machine learning and other types of analyses. The objective is to represent texts in a way that preserve their relationships and similarities within the content. The best representation of the documents will depend on the desired application, however the majority of processing of documents will rely on a distance measure in some way. For example, document classification and clustering can be done based on measures of distances between documents, calculated as a distance between vectors. This same approach is made of use in recommender systems. The whole idea behind recommender systems is to identify similar items, which can be thought of as a kind of distance measure between items. So in this case, techniques applied in document representation for classical machine learning purposes can also be assessed for effectiveness in representing text components in recommender systems.

The following section will provide some background on the two methods used in this project to generate text representations. This first is the word embeddings approach and the second is the tfidf and basic count vectors. These two approaches were used to generate document vectors, which were then used to identify similarities between documents. Document similarity is calculated as a distance measure, and the measure used in this project was the cosine distance, which will also be discussed.

3.2.1 Word Embeddings

From a free form text description of a movie, it is needed to generate a numerical representation of these words in some way, and a successful method of representing words for natural language

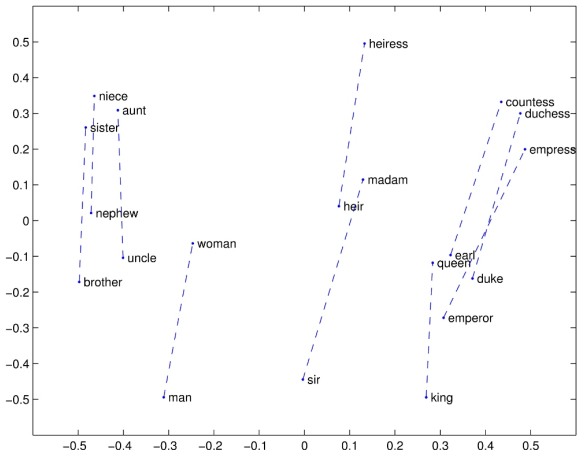

Figure 2: GloVe Relationship Representation [16]

processing studies is to use word embeddings. Word embeddings can be used to represent each word in the description as a vector. Then the full description vector can be calculated from these word vectors. These two main components, generating word vectors, and then generating document vectors from the word vectors will be discussed in the following section.

First of all, many different word embedding algorithms exist. Some popular algorithms include Google word2vec [12], and GloVe by Stanford nlp group [16]. The previous studies in this area [14] [15][13] used the google word2vec technique however because GloVe vectors have also been shown to produce good results in practice this project will focus on an implementation of the GloVe vectors. The GloVe algorithm is unsupervised and training is performed on aggregated global word-word co-occurrence statistics from a corpus. The GloVe model is log-bilinear with a weighted least-squares objective. The idea behind it is, that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. In the ratio of probabilities, noise from non-discriminative words cancel out, so that large values correlate well with properties specific relationships between words and the meaning associated with the concept of the word. The specific training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. The use of log enables the word representations to perform well on word analogy tasks as a ratio of logs, translates to a difference between two logs, and therefore resulting vectors can be added and subtracted in a linear way that preserves the meaning [16].

GloVe algorithm has been trained on large corpuses of text, such as Wikipedia, including 6 billion tokens, and the resulting representations have shown interesting linear substructures of the word vector space [16]. For example the GloVe vectors are capable of capturing the relations between woman and man, and have identified that this same relationship would be present between king and queen. Applying GloVe vectors enables representing words in a way that not only capture word similarities but also some of the semantic meaning such as the actual relations between the words. Not only does it provide some interesting relationship identification results, but it is also able to do this in a very low dimensional form of embedding vectors of lengths 50 to 300. For the purposes of this project, we will use the pre-trained GloVe vectors to generate representation of words. The smallest dimension of vector will be used as a point of comparison to previous studies that were using embedding vectors of higher dimension. It can provide some comparison to see if even in such low dimensions, can these word representations still enable capturing the required distances between words to generate good distance measures between documents.

Once the word vectors have been obtained from the text descriptions of movies, the next step is to generate a document embedding from a list of vectors (word embeddings). There are several ways to approach this that has been shown in past studies. The past study that is the main reference for this project [15] used the centroid of the word vectors to generate a document vector, so this is the method that will be used in this study. This is a common approach, however there are more recent advances that could enable an improvement on document representation. For example, the google word2vec algorithm has a modification to enable generating a doc2vec through the same training procedure - it would be interesting to assess this type of vector's performance in this application

for comparison as well. Also, another method called "sentence2vec" [9] takes into account word probabilities to generate a sentence vector from a list of word embeddings. This techniques, while promising, have not been assessed in this study, though would be great to add to the analysis for future research.

3.2.2 Count Vectors

One of the most basic ways to represent a document is just based on the count vector generated based on word occurrences. This can be created out of uni-grams bi-grams or trigrams, and the results is every term or n-gram that occurs in all the documents corresponds to a point in the array. This results in very larger dimension arrays where the weights correspond directly to term frequency. The result of this is that very frequent words will have much higher weights than other words, and therefore have a big influence on the results. This can negatively impact performance if preprocessing is not done to carefully removed the frequently occurring words that will just add noise to the evaluation. This method was assessed in this project as a simple approach to assess and compare performance. A more robust method that takes into account word specificity to certain documents will be shown in the next section.

3.2.3 Term Frequency Inverse Document Frequency (tf-idf)

Term frequency–inverse document frequency (tf-idf), is a statistical measure used very commonly in information retrieval. The tf-idf reflects how important a word is to a document in a collection. The tf-idf value for a word is weighted based on how many times the word occurs in a given document, but is inversely proportional to the frequency of that word in the corpus overall. The effect of this, is the weights will be proportional to how specific or important a word is within a specific document (i.e. the value is offset by the fact that some words are just very frequent overall). The value is calculated as: $tfidf(t, d, D) = tf(t, d)idf(t, D)$, where tf is the term frequency, and idf is the inverse document frequency.

This scheme is very popular- 83% of text based recommender systems use this method [4]. Therefore it is worthwhile including this representation technique in this investigation as a benchmark to use for comparison. Also, this technique will be assessed on the description of the movies as well as on metadata features to provide comparison between which of these features are able to best capture movie similarities.

3.2.4 Measuring Item Similarities

Now that we will have text features represented as vectors, how to assess the similarity between these vectors in a way that will represent similarities between movies? There are several possible approaches to measuring similarity between vectors.

Traditionally in linear algebra, the distance between two vectors is calculated as $d(\vec{u}, \vec{v}) = \|\vec{u} - \vec{v}\| = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 ... (u_n - v_n)^2}$, which is known as the euclidean distance. This distance measure relies the length of the vectors. However in text representation, vector lengths will be determined by word counts, and therefore longer documents will have longer vectors. This may not be the best way to measure differences between textual content, because for example, the vector "dog dog dog" may be less similar from the vector "dog" than the vector "cat" is similar to "dog". In representing text, the length of the documents may not best represent difference between content, and so alternative method of measuring distance between vector may be required.

Another method that can be used is called the cosine distance. This distance measure computes distance as the angle between vectors. This can be computed with the following equation: $\frac{x \bullet y}{\sqrt{x \bullet x}\sqrt{y \bullet y}}$. This is the dot product between two vectors normalized by the document lengths. It is clear that in this case, the document lengths do not come into play in assessing the similarity. This distance is not exact angle, but it represents a value that would be required to compute the cosine angle, and so the magnitude of this computation can produce the required comparisons between vectors. Also, note with pre-normalized unit length vectors, it is not necessary to compute the normalization when calculating this distance. This can lead to very efficient computation implementations, such as using the linear kernel function in python to compute a dot product between two vectors. Also, in non normalized vectors, the dot product can still be used as a distance measure, bearing in mind that the results will not be the same. If efficiency is required for calculations however, performing the dot product alone if it produces sufficient results in testing may

sometimes be used as the distance measure, though this will result in the distances being affected by the document lengths.

Another novel method that was recently developed is called the "Triangle's Area Similarity". This method purposely takes into account both the angle and the length of documents and uses this to compute the area of a triangle between two vectors. This has been shown to be a promising upcoming method of assessing similarity in text documents in a recent study by Heidarian, A. et al [7]. Another recent measure is known as the "word mover's distance" [11]. The word mover's distance is a measure that can be used to assess the dissimilarity between two text documents with word embeddings, and is calculated as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. These two more recent methods of assessing similarities have shown promising results applied to text, and would be interesting to assess in this application, however this will have to be left for future work.

4 Implementation and Experimentation

This section will explain the dataset used and the methodology applied. It will also briefly summarize the implementation, experimentations and observations, however much more detail on this can be seen in the accompanying file on the implementation, produced with Jupyter notebook.

4.1 Data Used

The dataset used is an enhanced version of the Movie Lens data set. The movies lens data set is one of the standard benchmark datasets used in evaluating recommender systems. This dataset enables comparing to other research on recommender systems that have implemented techniques on this data [1].

The dataset used was enhanced with metadata features and text descriptions, and was downloaded through the data science site "Kaggle". The specific dataset that can be found through Kaggle is called, "The Movies Dataset". Also accessed through Kaggle was an python notebook that was the starting point for much of this code, such as structuring the downloaded data into pandas dataframe, and implementation of some of the recommender functions. The specific notebook kernel used was called "Movies Recommender System" by Banik, R [3].

There are two sizes of datasets included, and experimentation was done primarily on the small dataset to allow quick testing and training throughout experimentation. This dataset consists of 100,000 ratings applied to 9,000 movies by 700 users. In the future, experimentation should include the larger dataset of 26,000,000 ratings to further assess the results.

4.2 Methodology

The methodology applied to generate recommendations for a specific user will follow the same procedure as described in the paper by Musto, C. et al. [15], as follows:

1. Given a set of items I, each i ∈ I is mapped to a Wikipedia page through a semi-automatic procedure. Next, textual features are gathered from each Wikipedia page and the extracted content is processed through a Natural Language Processing pipeline to remove noisy features.

2. Given a vocabulary V built upon the description of the items in I extracted from Wikipedia, for each word w ∈ V a vector space representation w_T is learnt by exploiting a word embedding technique T.

3. For each item i ∈ I, a vector space representation of the item i_T is built. This is calculated as the centroid of the vector space representation of the words occurring in the document.

4. Given a set of users U, a user profile for each u ∈ U is built. The vector space representation of the profile is learnt as the centroid of the vector space representation of the items the user previously liked

```
5. Given a vector space representation of both items to be recommended and user
profile, recommendations are calculated by exploiting classic similarity measures:
items are ranked according to their decreasing similarity and top-K
recommendations are returned to the user.
```

This same methodology was applied in this investigation as closely as possible. The main difference, is that instead of extracting the descriptions from Wikipedia, the descriptions for each movie were readily available through the Kaggle dataset. Also it is possible that there may have been differences in the text preprocessing pipeline. Specific details in the preprocessing applied in this investigation can be seen in the accompanying Jupyter notebook

Another main point to mention is the generation of a user profile from the data. There are many approaches that could have been used to perform this, however in keeping with the previous study the user profiles were generated in the same way as the the document vectors were created from the word embedding vectors, as previously described. The user profiles were created as a mean or centroid of all the movie description vectors they liked. This creates a vector to represent each user, and distance measures can be taken from movie vectors to each user vector to generate the list of similar movies for recommendations.

4.3 Experiments in Jupyter Notebook

Full details on the implementations of the recommender system and functions used can be found in the Jupyter notebook. This notebook will show:

- Loading and formatting data into pandas data frame
- Processing the text data into cleaned format (i.e. tokenization process)
- Implementation and evaluation of GloVe word embeddings
- Generating movie vectors:
  - Centroid of word embeddings
  - Tfidf of descriptions
  - Tfidf and count vectors of meta-data
- Assessing movie vectors: heuristic evaluations of similar movie lists
- Creation of user profiles from document vectors
- Generating recommendations based on a user profile vector
- Heuristic assessment of recommendations for users
- Comparison to Collaborative filtering
- Testing a hybrid approach
- Quantitative evaluation of results

Each of these steps will be shown in detail in the notebook, as well as some discussion on efficiency of implementation of python functions, and observations on results throughout the experimentation procedure.

5 Results and Discussion

The following section will briefly discuss the results obtained from the experimentation in the Jupyter notebook. First, evaluation methods for recommender systems overall will be discussed. Next, a discussion of the results based on these evaluation methods will be shown. Finally, a discussion of the overall methodology and possibilities for future work will presented.

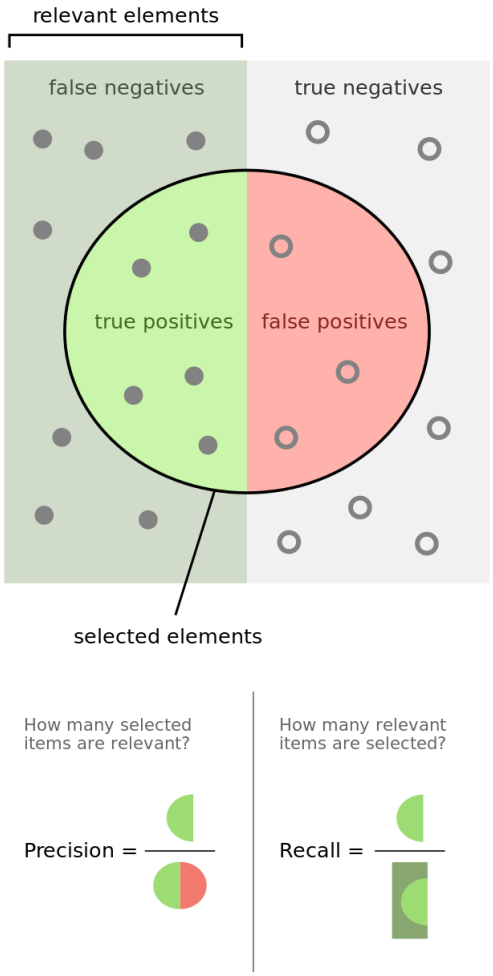

Figure 3: Precision and Recall Diagram [2]

5.1 Evaluation Methods

Evaluation is a big challenge in assessing recommendation algorithms. Some commonly used metrics are the mean squared error and root mean squared error, which are often used in assessing collaborative filtering recommenders. Information retrieval metrics are also commonly used, such as precision, recall and the F measure. Recently, diversity, novelty, and coverage are also considered as important aspects in evaluation, those these are not straight forward or standardized in how to assess. However, many of the classic evaluation measures are highly criticized as well [2]. Often, results that correspond well to these metrics do not actually correspond well to user satisfaction in practice. The being said, the methods of analysis chosen for this study are precision, recall and F measure. The methods are measured out the the top k results. This means that the top k, for example k being 10, results returned will be considered the positively predicted items [6]. The precision, recall and F measures are calculated based on a specific value of top k. Precision is measured as follows:

$$Precision = \frac{\{relevant-documents\} \cap \{retrieved-documents\}}{\{retrieved-documents\}}$$

Recall is measured as follows:

$$Precision = \frac{\{relevant-documents\} \cap \{retrieved-documents\}}{\{relevant-documents\}}$$

These two measures are summarized in the following image.

In both these equations, the number of retrieved documents will always be k (i.e. 10), so this is a fixed value that will not effect the results in comparing different models. Finally the overall measure called the F measure takes the harmonic mean of these previous scores. The result of this is the attempt to maximize both of them in the overall performance. The F measure is calculated

as:

$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$

Note how all these methods are highly dependent on features of the data. For example, when measuring the score for a user with a high number of rated items, for example 1000 items positively rated, the maximum recall for this user will be k/1000 for the top k results, so for example for k=10, the maximum recall for this user that is achievable is 0.001. Also, for precision, is a user only has rated 4 items positively, then the maximum precision is 4/10 even if all the possible items are included. This can have some strange impacts on the results, and make the produced value not intuitively interpretable. However these measures can be used to compare different recommenders tested on the same dataset.

There are many additional methods that could be used to improve the measuring of these scores [5]. For example, one technique is called cross validation, where the train and test sets are tested in every possible 80/20 split. Another method that seems very promising is call the leave one out method, where one ratings is left out and all other ratings are used to generate a prediction for that value of being either positive or negative.

5.2 Results

The following sections will show some of the results from each type of assessment done, though more can be found in the Jupyter Notebook.

5.2.1 Predicting Similar Movies

Many heuristic assessments were done comparing the results of predicting movies similar to a selected movie. The two main movies focused on were "Mean Girls" and "The Godfather". For detailed comparisons of these results please see the Jupyter notebook. A sample of the results is shown here for "Mean Girls".

Embedding method results:

Word	Cosine distance
Deadly Friend	0.954311
'Neath the Arizona Skies	0.949922
Phantasm II	0.944674
Another 48 Hrs.	0.943287
House II: The Second Story	0.942317
The Freshman	0.941801
Hello Mary Lou: Prom Night II	0.940905
Feeling Minnesota	0.940621
Crimes of Passion	0.940600

Description tf-idf results:

Word	Cosine distance
Wild Child	0.174726
Pitch Perfect	0.158836

```
                        The Craft 0.156621

                     Latter Days 0.145837

                      The Clique 0.140224

                Death at a Funeral 0.139270

                     Fallen Angels 0.136594

                     Doc Hollywood 0.135179

                  Mrs. Winterbourne 0.132972
```

Metadata tf-idf and count results:

```
                     titles    scores
3319         Head Over Heels  0.732575
4763           Freaky Friday  0.663749
7905  Mr. Popper's Penguins  0.663324
6277        Just Like Heaven  0.644200
1329        The House of Yes  0.641305
```

The main conclusions from these heuristic movie similarity judgments is that all the methods produced some reasonable seeming recommendations, however, the tf-idf and count methods on the metadata seemed to perform the best overall for most of the movies, with the tf-idf description method close behind. The embedding method produced some reasonable sounding movies, such as the listed movie "Hello Mary Lou: Prom Night 2" which makes sense that it would have had similar high school based content description. However some of the recommendations from the embeddings method seemed quite off track. Also, note that all the similarity scores from the embedding method were very close. Perhaps the distance measure being used does not distinguish between similar movies well enough, and is returning many movies with high similarity ratings.

5.2.2 Recommending Movies to Users

Next we performed some recommendations to users based on the generated user profile vectors. Once again the tfidf method with the metadata seemed to perform the best. A sample of the results is as follows:

```
titles scores
1251 The Lost World: Jurassic Park [0.370625966193]
1497 Armageddon [0.356943476739]
5869 Twilight Zone: The Movie [0.346949005222]
1062 Indiana Jones and the Last Crusade [0.344380646326]
6232 War of the Worlds [0.341673363712]
972 Raiders of the Lost Ark [0.340651711682]
427 Jurassic Park [0.33946901943]
1241 The Fifth Element [0.329456655949]
232 Star Wars [0.327395220779]
2120 Star Wars: Episode I - The Phantom Menace [0.326242953738]
1498 Lethal Weapon 4 [0.324961021804]
1580 Lethal Weapon 2 [0.320809011678]
983 Return of the Jedi [0.312446308709]
2788 Close Encounters of the Third Kind [0.31016389337]
3674 Planet of the Apes [0.303785575676]
2043 Planet of the Apes [0.302834432877]
7024 Indiana Jones and the Kingdom of the Crystal S... [0.300160146696]
6242 The Island [0.297540370567]
1692 Indiana Jones and the Temple of Doom [0.29349220226]
522 Terminator 2: Judgment Day [0.291468118107]
```

The movies the user had selected as liked are: 'Braveheart', 'Apollo 13', 'Star Wars', 'Jurassic Park', 'The Silence of the Lambs', 'Die Hard 2', 'Mars Attacks!', 'Batman and Robin', 'Lethal Weapon 2', 'The Mask of Zorro', 'The Mummy'. This shows really great results because it predicted all the movies the user had liked in the top ten. So extending the list to check the next recommendations, these also look like quite reasonable fitting recommendations given the movies the user had liked.

The results of the predictions from the tfidf method and the embedding method were also compared to the results of collaborative filtering. Collaborative filtering was used to generate predictions for the user's unrated movies. Movies from the top ratings list from both methods were assessed using the collaborative filtering rating prediction, and both movies were predicted to have a good rating of approximately 4/5. This is a good sign that even the embedding method is capable to return movies that would have a decent rating by the user in its top ten list of results.

5.2.3 Hybrid Method

A hybrid method was experimented with as well. This method was shown to be able to provide movie recommendations based on collaborative filtering predicted scores, as well as by taking into account a movie the user was interested in. This is an interesting way to combine both the content based recommendations and the stable well validated approach of providing recommendations with collaborative filtering. This could be applied by having a user select a movie they are interested in, and the recommender providing recommendations of similar content based on the users previous movie ratings. Results are given in Jupyter notebook for a particular user, for the two cases of the selected movie being "The Godfather" and "Mean Girls", and they seemed like reasonable results.

5.2.4 Quantitative Evaluation

A quantitative evaluation was done to perform an assessment of the F measure, precision and recall as previously described. This was computed several different ways in order to find the approach that could be compared with the previous study. First the values were calculated by adding up the total number of correct predictions across all users and the total number possible positive results across all users, and these values were used to produce the measures. This was repeated again by applying a trained model on all the test data to produce a precision and recall score for each user, and the averages were taken to produce a final value for all users, and these values were used to compute the F measure. The results were similar in both cases and much lower than values found in the literature.

The F measure calculation was modified to only test on results each user rated in the prediction of the top k results, which therefore greatly increases the odds of getting a correct result in the top k. The result of this was a significant increase in the values for each of these measures, which are now comparable to the literature. Actually, the values produced by these techniques actually slightly outperformed the values in the study by Musto et. al [15], but further testing should be done on the larger dataset and with cross validation to confirm this conclusively. A summary of the results at the top 10 is shown below:

```
Tf-idf meta-data results:
F1:  0.6192

Count vector meta-data results:
F1:  0.6106

Musto et al results (word2vec, 300 dimension)
F1: 0.5757

Musto et al study results (word2vec, 500 dimension)
F1: 0.5751
```

5.3 Discussion and Future Work

So far the heuristic assessments of the results seemed to provide reasonable recommendations, with tfidf on the meta-data as the best results. It seem the content in the key words and cast

and crew members has a stronger association with which movies are similar that the actual movie descriptions.

This investigation only scratched the surface on the experimentation that could be done on this, and there is much room for future work to be done. There is much more experimentation and investigation that could be done with the text representation, with the distance measures, and with the evaluation methods.

First of all, for testing different word embeddings, there are numerous newly developed models that could be test. For example, Facebook Fasttext and LexVec and two more recent word embedding models that could be investigated for this application. Also, experimentation could be done on pre-training these embeddings on different dataset. Another very worthwhile investigation would be to compare the results of embeddings of different dimensions, to see how the size of the embedding relates to performance.

These methods could also be extended to apply to test on the meta-data words, and incorporate the meta data into the vector as an embedding, as was done in the previous study by Kim et al [10]. The feature vectors could also be concatenated to investigate incorporating both tfidf vectors and embeddings, and this could be assessed with feature selection techniques. Finally the vector representations could also be tested with added additional non-text features, such as overall popularity of the movie, amount of money earned, etc.

Next, there could be much more investigation done on the distance measures used. As mentioned previously new promising metrics in document distance measurements have emerged such as the word movers distance, and the triangle area measure. Assessing these as metrics of similarity instead of the cosine distance would be a very interesting analysis to add to this investigation.

Finally, more testing could be done in order to assess the evaluation methods themselves. In order to tune the performance of these recommendations, it would be worthwhile to investigate different ways of assessing the results, to truly determine the quality of performance. This is tricky in this type of situation, where the objective is to provide the user recommendations to unseen items, where it is unclear exactly how much they would have liked that item. Therefore to evaluate conclusively evaluations should be done on only the items the user has rated, but in that case, some modified evaluation approaches could become of use. For example, the leave one out method individually assesses each rating if the user would have predicted it as negative or positive given all the other data in the model. This approach would take care of the issues that some users would have more or less ratings in the test/train set, as each rating will be predicted individually. Because this method was not used in past studies this was not assessed in this investigation however it would be interesting to perform some comparisons of models using this method of evaluation.

The results of the F measure evaluations showed that even the standard count vector document representation outperformed the result from the previous study however, as mentioned some very important next steps that should be taken to further assess this is to study the effects of the dimension size of document vectors on the results.

6 Relevance and Conclusions

The main conclusion from these results is that the small dimension word embedding method with GloVe vectors seemed to not properly capture similarities between movies well, and was outperformed by the tfidf methods. In particular incorporating meta-data feature produced very good results. The heuristic evaluations all showed fairly reasonable lists. The quantitative results showed that these tfidf representations perform well for completely unsupervised recommendations compared to the literature. Overall these methods are promising and can definitely be used to produce viable recommendations as a stand alone method , and also in conjunction with other classical recommendation methods. The relevance of this analysis is that it provides additional insights into how to represent text features for recommendation systems that can be built upon to lead towards incorporation into larger more complex models such as hybrid systems.

7 References

References

[1] Kaggle the movies dataset, 2017.

[2] Wikipedia precision and recall, 2017.

[3] Rounak Banik. Kaggle movie recommender systems, 2017.

[4] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[5] Alejandro Bellogín, Pablo Castells, Alan Said, and Domonkos Tikk. Workshop on reproducibility and replication in recommender systems evaluation: Repsys. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 485–486. ACM, 2013.

[6] Tommaso Di Noia, Iván Cantador, and Vito Claudio Ostuni. Linked open data-enabled recommender systems: Eswc 2014 challenge on book recommendation. In *Semantic Web Evaluation Challenge*, pages 129–143. Springer, 2014.

[7] Arash Heidarian and Michael J Dinneen. A hybrid geometric approach for measuring similarity level among documents and document clustering. In *Big Data Computing Service and Applications (BigDataService), 2016 IEEE Second International Conference on*, pages 142–151. IEEE, 2016.

[8] Chen Hung-wei, Yi-leh Wu, Maw-Kae Hor, and Cheng-Yuan Tang. Fully content based movie recommender system with feature extraction using neural network. In *Proceedings of the 2017 International Conference on Machine Learning and Cybernetics*, pages 9–12, 2017.

[9] Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*, 2017.

[10] Sung Min Kim and Young Guk Ha. A new method of measuring document similarity for movie recommendation. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on*, pages 41–44. IEEE, 2014.

[11] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[12] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[13] Cataldo Musto. Enhanced vector space models for content-based recommender systems. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 361–364. ACM, 2010.

[14] Cataldo Musto, Giovanni Semeraro, Marco De Gemmis, and Pasquale Lops. Word embedding techniques for content-based recommender systems: An empirical evaluation. In *RecSys Posters*, 2015.

[15] Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. Learning word embeddings from wikipedia for content-based recommender systems. In *European Conference on Information Retrieval*, pages 729–734. Springer, 2016.

[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[17] Alexandre Salle, Marco Idiart, and Aline Villavicencio. Matrix factorization using window sampling and negative sampling for improved word representations. *arXiv preprint arXiv:1606.00819*, 2016.

[18] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge, 2017.

[19] Haiming Wang, Peng Zhang, Tun Lu, Hansu Gu, and Ning Gu. Hybrid recommendation model based on incremental collaborative filtering and content-based algorithms. In *Computer Supported Cooperative Work in Design (CSCWD), 2017 IEEE 21st International Conference on*, pages 337–342. IEEE, 2017.