Machine Learning Engineer Nanodegree

Capstone Proposal

Tyler Shumaker
July 30, 2017

Proposal

Domain Background

I am currently working on a project to develop a proof of concept implementation of intelligent virtual agents that will provide virtual personal assistant, recommendations, and product socializing capabilities to an user. The users will interact with the intelligent agents via chat, using an open source webchat platform as the interface to the chatbot. One of the jobs being performed in the background by the intelligent agents is to provide recommendations to the users. We intend to recommend chat rooms to users they should consider joining.

Providing recommendations based off of chat discussions has been used to provide complementary information. Recommendation of Complementary Material during Chat Discussions [1] provided recommendations of electronic documents and links to web pages to enable collective and personal learning. This proposed system performed text mining on each message posted to a chat program and compared words in the message to terms in a domain ontology. Terms identified in the message were then passed to a Recommender module where items classified with those terms were then recommended. This proposed system is similar to what we are trying to achieve but we will be recommending chat rooms that represent an area of interest. Ex. Chat room titled "Ocean City, MD" will be used for users to chat about Ocean City, Maryland. So users chatting about terms like "Ocean City" and "Maryland" will be recommended to join the "Ocean City, MD" chat room.
* Details of the project are intentionally vague in this proposal because the work is on-going.

Problem Statement

A common problem for our clients is that user's spend a lot of time finding a subject-matter expert (SME) or relevant information to assist them complete their tasks. Based upon the user's interaction with the chatbot and other users we want to be able to recommend chat rooms where other users are working on similar topics. Users are able to chat with other users in public chat rooms, private chat rooms and direct messages. No communication via private chat rooms or direct messages will be used for recommendations. Users can also interact with the chat bot via direct message and in public chat rooms by starting a message with the chat bot's handle. Ex. @chatbot

All communication between a user and the chat bot will be captured as conversations because the system is designed for the chat bot to perform actions for the user that are relevant to their tasks. Ex. @chatbot search google for "vacation destinations in Maryland" which the chat bot will respond

with a list of links to the top 10 responses of a Google search of the search term. Continuous chat exchanges (separated by 30 minutes of inactivity) between multiple users in a public chat room will be recorded as a conversation. The system allows for chat rooms to have titles ("Ocean City, MD") but also associated topics of interest such as "Vacation", "Ocean", and "Maryland" to that chat room. Users will be given recommended chat rooms at predetermined intervals (daily) or when requested from the chat bot (@chatbot get recommended chat rooms). The recommendations will be returned by the chat bot via a direct message to the user.

A future feature of this system could be an explainability feature. So that the users can trust in the intelligent agent's recommendations a degree of explainability needs to be provided. When the recommendations are provided to the user they will be given in the following format; Chat Room "X" is recommended because users are also discussing \. Ex. Chat Room "Ocean City, MD" is recommended because users are also discussing ["Ocean", "Maryland", "MD"]. This feature will not be covered in this project.

Datasets and Inputs

The Cornell Movie-Dialog Corpus [2] will be used because it is a collection of conversations extracted from movie scripts. The dataset consist of 220,579 conversational exchanges between 10,292 pairs of movie characters involves 9,035 characters from 617 movies. Each character can be treated as a user in a chat platform and each movie can be related to chat rooms or channels. The movie scripts were gathered from publicly available scripts that are referenced in the raw_scripts_urls file. The metadata for the movies were gathered from IMDB. The corpus consist of a movie_titles_metadata file that provides; movieID, movie title, movie year, IMDB rating, no. IMDB votes and genres. The movie_characters_metatadata, movie_lines and movie_conversations files can be can be used to construct the conversations between movie characters.

Characters from the movie corpus are equivalent to users in our chat system and movies will be equivalent to chat rooms. With the movie corpus data I will be recommending movies (chat rooms) to movie characters (users). In our application the users and chat rooms can be associated with topics of interests which will be managed by users. The associated topics of interest are not a reliable means of providing recommendations because it requires the users to populate this data. Users may not associate any topics or incorrectly assign topics to chat rooms.

To determine the similarity for movies all of the lines from that movie will be processed as a data entry. The same algorithm will be applied to all the conversions for each character. By comparing the similarity of a character conversation to the movie scripts (details in solution statement) a recommendation wil be generated by the system.

Solution Statement

All of the movie lines from each movie will be converted to vectors using a word embedding model, Doc2Vec (specifically the “distributed memory” (dm) algorithm) [3]. "Doc2vec (aka paragraph2vec, aka sentence embeddings) modifies the word2vec algorithm to unsupervised learning of

continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents." [4] The movies from the Cornell Movie-Dialog Corpus will be used to train the model and the conversations of individual characters will be used to test the model. To assess the model the training corpus will be compared to inferred vectors from the training corpus and return the rank of the document based on self-similarity. [5] It should be easy determine if the model is performing correctly if a vector for a movie returns the highest rank for itself (if a character is from "movie A", then "movie A" should return a high rank). The model then be tested against random movies and characters from the test corpus and compared by eye. The end of the system will return the top movies 5 movies returned from character conversation inferred vectors.

Benchmark Model

A content-based recommendation system will be used a benchmark model. TF-IDF (Term Frequency - Inverse Document Frequency) will be used to parse all of the conversations for each character and also all of the conversations for each movie. Cosine similarity will be used to determine which conversations are closest to each other. [6] The movies that the highest cosine similarity to a characters conversation will be recommended. The recommendation of the Doc2Vec approach will be compared to the results of the content-based recommendation system by comparing the text of the of the recommended movies by eye.

Evaluation Metrics

To evaluate this solution the similarities in recommendations for a character (A) and a character that has a conversation with character (A) in a movie (B) will be compared to the to the similarities in recommendations for character (A) and a random character (C). Ex. The recommendations for "Indiana Jones" and "Professor Henry Jones" (Indiana's father in "Indiana Jones and the Last Crusade") should be more similar then the recommendations for "Indiana Jones" and the "Juno" character from the movie "Juno".

Project Design

The Cornell Movie-Dialog Corpus will need to be preprocessed so that conversations can be utilized in the word embedding model. The Gensim [7] pre-processing utility will be utilized to convert the movie scripts and individual character conversations into a list of tokens (individual words, remove punctuation, set to lowercase, etc). The movie id from the Cornell Movie-Dialog Corpus will be used as a tag for each movie document. The individual character conversations will be tagged with the character id from the Cornell Corpus. The Doc2Vec model will be initiated with a vector size of 30 words and iterating over the training corpus 20 times.

```
model = gensim.models.doc2vec.Doc2Vec(size=30, iter=20)
```

Different vector size, number of iterations and minimum word count (will give higher frequency words more weight) will be tested to see if increases in those parameters improve results while assessing the model. To do the assessment of the model the steps used in the "Doc2Vec Tutorial on

the Lee Dataset" [5] will be utilized. This will required to infer new vectors for each movie of the training corpus, compare the inferred vectors with the training corpus and then return the rank of the document based on self-similarity.

```
ranks = []
second_ranks = []
for doc_id in range(len(train_corpus)):
    inferred_vector = model.infer_vector(train_corpus[doc_id].words)
    sims = model.docvecs.most_similar([inferred_vector], topn=len(model.docvecs))
    rank = [docid for docid, sim in sims].index(doc_id)
    ranks.append(rank)

    second_ranks.append(sims[1])
```

The different model parameters will be tested here to see if the percentage of inferred documents found to be similar to itself can be increased.

The model will then be tested with randomly chosen character conversations from the test corpus. It would be expected that the movies that a character is from should be the most similar movie. If undesirable results are found during this testing the model may have to be revisited and assessed over again. Further testing will be done comparing the results of characters and characters that have had conversations with the test character. The most similar results for these two characters should be more aligned then results of that character and a randomly chosen character.

After the testing as resulted in a model that is satisfactory a content-based recommendation system will be utilized as a benchmark model. The benchmark recommendation system will use the same training corpus (movie lines) that was used to train the Word2Vec and testing corpus (character conversations) to train. TF-IDF will parse through all of the training text looking at one, two and three-word phrases that appear multiple times in the text and divide by the number of times the same phrases appear in all of the training data sets. Common english words will be ignored using a 'stop_words' parameter.

```
        tf = TfidfVectorizer(analyzer='word',
                             ngram_range=(1, 3),
                             min_df=0,
                             stop_words='english')
```

SciKit Leanr's linear_kernal will be used to compute the similarity between all test data sets. The predict method will return the top five movie results as indicated by the "m*" id.

The same characters will then be tested against the Doc2Vec model and the TF_IDF model. The accuracy of each model will be judged by the percentage of results that return the movie which the character is from as the most similar.

References

[1]Loh, S., Lichtnow, D., Kampff, A. J., & Moreira de Oliveira, J. P. (2010). Recommendation of Complementary Material during Chat Discussions. Knowledge Management & E - Learning: An International Journal, 2(4), 385-399. Retrieved July 11, 2017, from http://www.kmel-journal.org/ojs/index.php/online-publication/article/viewFile/20/63

[2]Danescu-Niculescu-Mizil, C. (2011). Cornell Movie--Dialogs Corpus. Retrieved July 6, 2017, from http://www.cs.cornell.edu/~cristian//Cornell_Movie-Dialogs_Corpus.html

[3]Quoc V. Le, and Tomas Mikolov, Distributed Representations of Sentences and Documents ICML, 2014 from https://cs.stanford.edu/~quocle/paragraph_vector.pdf

[4]Rehurek, R. (2014) Doc2vec tutorial. Retrieved July 18, 2017, from https://rare-technologies.com/doc2vec-tutorial/

[5]RaRe-Technologies (2017, June 10), Doc2Vec Tutorial on the Lee Dataset. Retrieved July 18, 2017, from https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/doc2vec-lee.ipynb

[6]Clark, C. (2016, June 09). A Simple Content-Based Recommendation Engine in Python. Retrieved July 6, 2017, from http://blog.untrod.com/2016/06/simple-similar-products-recommendation-engine-in-python.html

[7]Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. THE LREC 2010 WORKSHOP ON NEW CHALLENGES FOR NLP FRAMEWORKS, 45-50. Retrieved July 17, 2017, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.695.4595