AIsha - Conversational Movie Recommender

Anusha Mohan, Anjali Unnithan
W 266: Natural Language Processing
UC Berkeley School of Information
{anushamohan, aunnithan}@ischool.berkeley.edu

April 2021

Abstract

Within recent years, BERT has been universally considered as instrumental to transformer architectures, especially for its high accuracy in Natural Language Processing tasks, such as sentence classification and sentiment analysis. This project explored the incorporation of the BERT transformer architecture in assisting with the task of intent classification in a recommendation system. Our project is heavily inspired by the existing work done in the movie chat bot realm, but many focused predominantly on QA chatbots. Our goal was to not only create a recommender, but also make a conversational chatbot that would only give recommendations when triggered using intent classification. From our baseline model that incorporates a deep neural network, our average F1 score amongst all intent classifications jumped from a 0.34 baseline to a 0.97, a 63 percentage difference, revealing the high efficacy of BERT in classification tasks that helped to streamline the conversation bot.

Introduction

In a world that is becoming more focused on personalization to deliver the ultimate user experience, recommender systems and engines have risen in popularity, be it through streaming services or advertising. Movie recommendation is a common task that is being constantly iterated and improved on a larger scale in companies like Netflix and Hulu. On a smaller scale, our product, AIsha, is an NLP based chatbot that recommends highly rated movies for users based on their preferred genre, cast, and topical preferences. Based on this paper [1], we see that individuals who interacted with a chatbot compared to qualtrics respondents significantly increased participant engagement. Due to this, rather than building a regular movie recommendation bot, incorporating a conversational aspect will boost engagement and enjoyment. AIsha remembers the user's preferences throughout the conversation and continually provides recommendations until the user is satisfied. In this paper, we discuss our approaches and algorithms for creating a movie recommendation system, specifically a content-based filtering system, as well as our process to achieve the intent classification task.

Background

Previous work done in the space of topic-aware chatbots incorporate recurrent neural networks and non-negative matrix factorization [2]. This was achieved by combining the traditional RNN structure with a topic-aware layer. This paper was a huge inspiration for our project, as we wanted to take a traditional BERT infrastructure and use the intent recognition as our form of topic-awareness.

Systems used frequently in the film recommendation setting include content-based filtering, memory-based collaborative filtering, model-based collaborative filtering, and recommendation through a deep neural network approach. For our recommendations, we focus on the content-based filtering approach, in which our user will be presented with films based on similarity and proximity to their indicated preferences. We believed this approach would be ideal especially because we did not have data on multiple users and their movie preferences. The similarity metric of our choice was cosine similarity, applied after all word soup vectors were created. For our baseline approach, we converted our intent file into "documents" that were then converted into a bag of words framework. Our next step was to build a deep neural network (DNN). We highly referenced this article [3], which discusses creating chat bots with tensorflow and DNN.With this DNN, we created a tflearn model that can then be run on the transformed data. Figure 1 is a diagram of our DNN baseline flow. Our goal for the baseline was for the network to classify the intent of the user input, which would then be a class in our chatbot. We created a function that would run the tflearn model.predict on the data, and return the probabilities associated with each intent. Based on the highest probabilistic intent, it would classify the user input. The DNN approach had a very low accuracy with an f1 score of 0.34.

Therefore, we made a decision to transition from

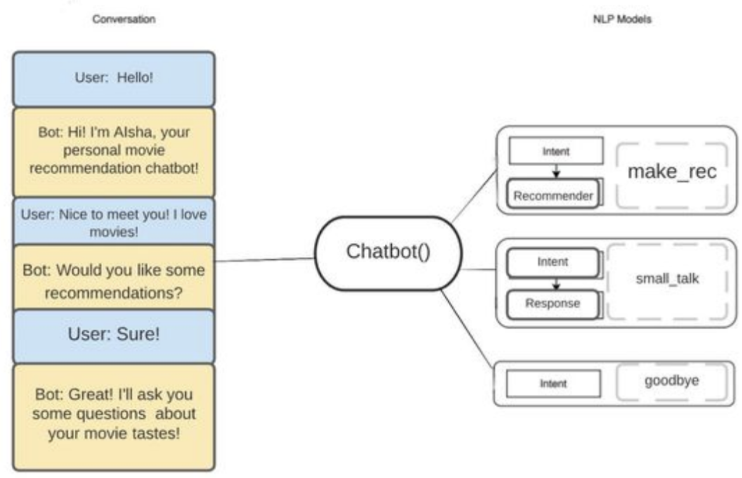


Figure 1: DNN Architecture

DNN to incorporating BERT BASE due to BERT being pre-trained on a large corpus. This provides an advantage in classifying certain word phrases associated with chatbot intents. Our current approach also incorporated a more robust intent data file. The intricacies of our current approach will be detailed in the following sections.

When incorporating BERT into the intent classification task, we referred highly to the code given in this article [4]. We specifically incorporated the uncased BERT-Base, which involves 12 layers, hidden size of 768, and 12 self-attention heads. We loaded the BERT model directly from the Google API [5].

The BERT model had several files, including configuration ones and a vocabulary file, which was loaded and tokenized. Afterwards, we preprocessed our intents dataset, split into train and test, and fed the input training data (i.e. the possible user phrases) into the pre-trained BERT model to get embeddings for the classification task. The [CLS] and [SEP] tokens get included and padding for shorter input texts are also added. A Keras model that includes a BertModelLayer based on the BERT configuration is then created. The remaining layers are appended, including hidden layers, dropout layers, and the final layer being a softmax one. The model is then compiled, and train data (80 percent of the entire intents file) is then fitted. Afterwards, we evaluate on the test data (20 percent of the entire intents file) . For the remainder of the paper, we refer to the post-baseline approach as the "BERT approach".

Here are some of the research questions we considered: On which intent classes does BERT consistently perform well? Which intent classes have lower scores, and why does the frequent misclassification occur? Additionally, what would be the best way to demonstrate a smooth chatbot flow? How do we incorporate all special cases and directions of the user-chatbot conversation? For example, we would have to consider a case in which the user may ask for additional recommendations if not satisfied with the first few ones, as well as if a user was more interested in small talk and informally conversing with the chatbot with topics other than films. How effective and appropriate would the generated responses be?

Project Overview

There are two components to this project, one being the creation of a movie recommendation algorithm and the second one being the BERT-Base trained on an intents dataset that we created. For the movie recommendation, we gather word soups for each individual movie from the Movies Database dataset, consisting of metadata of approximately 45,000 movies released on or prior to July 2017. We highly referenced this article to create the movie recommendation system [6].The word soups consist of the movie title, key actors/actresses, genre, and descriptive keywords, such as "funny", "heartfelt", and "violent". The soups are then tokenized, vectorized, and within the recommendation algorithm, cosine similarity scores are calculated after the user's preferences are transformed into a word soup of its own. The top 10 recommendations based on similarity scores are then generated for the user's viewing. As for where BERT comes into play, we have created an intent dataset with two columns: one being a user dialogue phrase, such as "What's up" and the second being the intent class, which in this case is "greeting". The pre-trained BERT model is trained on this intents dataset.

Prior to incorporating BERT for the intent classification task, we had a baseline in which our intents file was in json format. All user "patterns" were through the stemming and tokenization process, after which we had 8 intent classes ["I've already seen these movies", 'goodbye', 'greeting', 'no', 'make rec', 'small talk', 'thanks', 'yes'] and 95 unique stemmed words. Data was then split into train and test, and a bag of words function was created to convert an input sentence into an array of 0s and 1s, so that the DNN model could perform the classification task. Using tensorflow, we created a deep neural network with two fully connected layers and an output layer. After compiling the model on 1500 epochs, losses of around 0.02 and accuracies of around 0.98 were reported. Despite the impressive results, when actually manually testing random phrases not included in the original dataset, we found poor classification results, with phrases like "I am in the mood to watch a comedy film" being classified as "goodbye".We created a classify function that takes in the user input and uses the model to return the predicted intent classification. Once we have this information, our response function takes in the predicted intent from the classify function and returns a response from the bot based on what our model predicted the user's intent was. This is the crux of the conversational aspect of our project. From this, if a movie recommendation intent is predicted, it will trigger our recommendation system which will work with the user till a satisfactory recommendation is given. We believed we could find significant improvements in accuracy if we were to use BERT for model training instead. We tweaked the intent classes to include 'give additional recs' and 'about me', where a classification of 'about me' on a user input would prompt the chatbot to describe details about itself to the user.We did a 80 to 20 train to test split on our data so that we could test our model on the test data after training using BERT. Another resource that we found inspiration in is this paper that talks about creating an AI based chatbot for hotel bookings and the theory behind task-oriented chatbots [7]. Their approach heavily incorporates intent classification and extracts information that a user mentions during the conversation to recommend potential hotels. This paper was very influential in our project, in that we hope to "store" the information behind user's film preferences to give them ideal recommendations based on high similarity scores. Similarly, their AI chatbot has predetermined intents such as "thanks", "stop", and "cancel" and they heavily incorporate BERT for classification. However, the main difference between our approach with theirs is that they use SpaCy trained NER models and LSTMs in their neural classification models.

Methods

In terms of methodology, our project focused on intent classification based on user's input. The intent classification is used for the conversational aspect of the bot; Based on the user's input, our model should take in the input, classify it based on our intent classes, and return that class; through this, the bot would respond with an appropriate response based on the predicted class. One of the intents triggers our movie recommendation system, which uses metrics such as cosine similarity to grab the best movie recommendations based on user input.

Model: "model_1"

Layer (type)	Output Shape	Param #
input_ids (InputLayer)	[(None, 14)]	0
bert (BertModelLayer)	(None, 14, 768)	108890112
lambda_1 (Lambda)	(None, 768)	0
dropout_3 (Dropout)	(None, 768)	0
dense_8 (Dense)	(None, 768)	590592
dense_9 (Dense)	(None, 9)	6921

Total params: 109,487,625
Trainable params: 109,487,625
Non-trainable params: 0

Figure 2: Model Parameters

Design

Our project design is bi-directional; We first began by implementing our movie recommendation system, which uses the movie-corpus metadata with about 40,000 rows of different movies with information including genre, actor, director, ratings, and keywords. Due to sizing constraints, we were only able to add a subset of the overall movie dataset. We then created a soup that takes in important descriptive words about the film and puts it into an all encompassing narrative that could adequately describe the film. With this information, we used cosine similarities to map similarities between soup values. With the dataset ready, various functions that grab information from the dataset such as actor, director, and genre are used based on the user's input. For the conversational portion of our project, we started with a baseline that used deep neural networks that was trained on our intents dataset. With this baseline, our accuracy was quite low, with the correct intent classification being missed with even slight variations in user input. To improve our accuracy, we shifted to a BERT based architecture that was already pretrained. We then increased the size and length of our intents file, along with its structure. Our design architecture can be seen in Figure 3.

Our Implementation (Model)

With BERT parameters, we created a model trained on our intents dataset with the features in Figure 2.

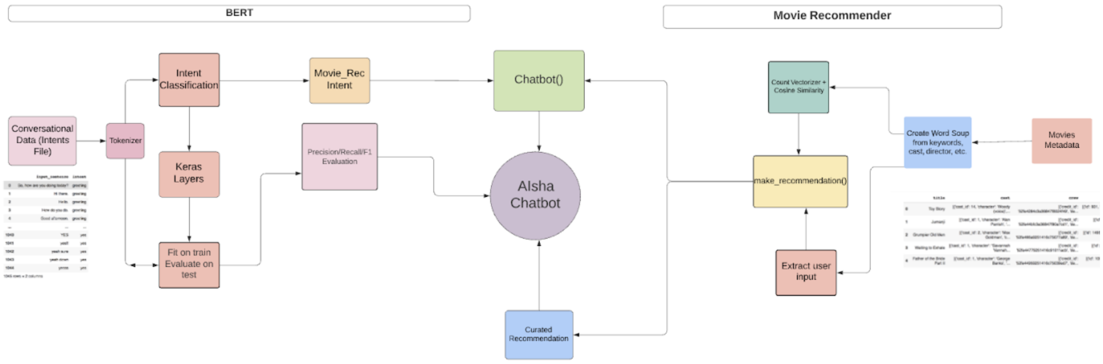


Figure 3: Recommender Bot Architecture

Results

We were significantly able to increase the accuracy of our model and intent classification when we shifted to the BERT architecture compared to only using deep neural networks. From DNN to BERT, we see an increase in accuracy from 34 to 98 percent, an increase by 64 percent. See Figure 4 for our classification report for our intent classification with BERT. There are multiple reasons for which this increase was so drastic. For the DNN model, the size of the intents file is smaller; as for each intent class, there were a limited number of possible phrases, whereas for BERT, our dataset was larger and had a variety of phrases for each class. Moreover, including additional layers and neurons into the network for the BERT approach improved accuracy by 1 percent; the train-test split was done to incorporate a variety of intent classes so that there isn't an issue of over or under sampling. Going from CountVectorizer to TfidfVectorizer and increasing the number of movie rows in our data from 15000 to 20000, as well as only considering highly rated movies, greatly improved the quality of recommendations given. A better vectorizer and more data made the recommendations more appropriate and fitting based on user preferences for keywords, genres, and movie stars.

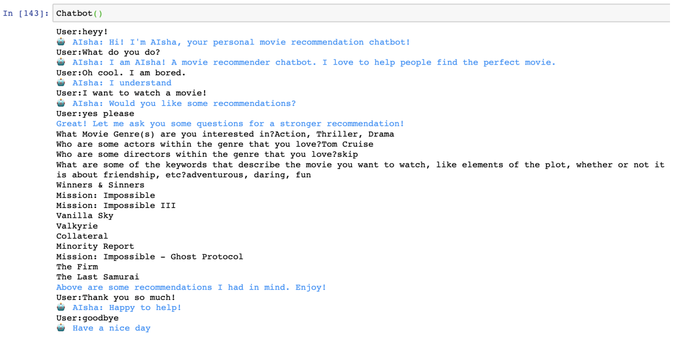
Evaluation and Error Analysis

Despite extremely high precision, recall, and F1 scores for the BERT approach, evaluating the chatbot also went beyond intent classification metrics. We also ran our chatbot function multiple times to experiment with various user responses and conversation topics. For example, some conversations incorporated small talk, such as asking about the weather or what AIsha's favorite food is. Technically, it was correct identifying the user inputs as "small talk", and AIsha responded appropriately by reminding the user that it is a movie recommendation chatbot, but realistically, the user would expect a chatbot to respond more elaborately.

On the movie recommendation side, the original movies dataset contained 40000 rows approximately, but to avoid the notebook file crashing, we had to limit the scope to 20000 rows. The resulting recommendations from the chatbot would've possibly been stronger and more sensible if the full dataset had been incorporated. Having keywords in the word soup certainly helped, but assigning weights to certain features like the main cast would probably have given more familiar recommendations. For example, when giving "Leonardo Dicaprio" as a user input and "boat", "ship", and "love" as keywords, "Titanic" was not one of the recommendations given. Instead there were more Leonardo Dicaprio movies suggested. In addition, when just giving the input "dicaprio", none of his movies showed up. Perhaps in a future iteration, BERT could be used in the recommendation aspect as well, where word soups would be the input fed into the training data along with its label being the respective film.

There were particular user inputs that were misclassified and therefore the respective chatbot response was not appropriate. For instance, the chabot tends to misclassify greeting inputs and confuses them with goodbye inputs. This is also explained by the lowest recall score of 0.80 under "goodbye", and may be due to the similarity in length of the word phrases like 'bye', 'hello', 'hi' and also its context.

Another occurrence of misclassification was with the user input after they received their recommendations. Phrases like "i'll be sure to check these out" should be classified as "thank you", since a user is responding to the receiving of movies. However it gets misclassified as "give additional recs". The word choices of the user input doesn't make it clear that it is a sign of appreciation or gratitude for the recommendations. However, another possible reason



```
In [143]: Chatbot()
User:heyy!
🤖 AIsha: Hi! I'm AIsha, your personal movie recommendation chatbot!
User:What do you do?
🤖 AIsha: I am AIsha! A movie recommender chatbot. I love to help people find the perfect movie.
User:Oh cool. I am bored.
🤖 AIsha: I understand
User:I want to watch a movie!
🤖 AIsha: Would you like some recommendations?
User:yes please
Great! Let me ask you some questions for a stronger recommendation!
What Movie Genre(s) are you interested in?Action, Thriller, Drama
Who are some actors within the genre that you love?Tom Cruise
Who are some directors within the genre that you love?skip
What are some of the keywords that describe the movie you want to watch, like elements of the plot, whether or not it
is about friendship, etc?adventurous, daring, fun
Winners & Sinners
Mission: Impossible
Mission: Impossible III
Vanilla Sky
Valkyrie
Collateral
Minority Report
Mission: Impossible - Ghost Protocol
The Firm
The Last Samurai
Above are some recommendations I had in mind. Enjoy!
User:Thank you so much!
🤖 AIsha: Happy to help!
User:goodbye
🤖 Have a nice day
```

Figure 4: Example Conversation

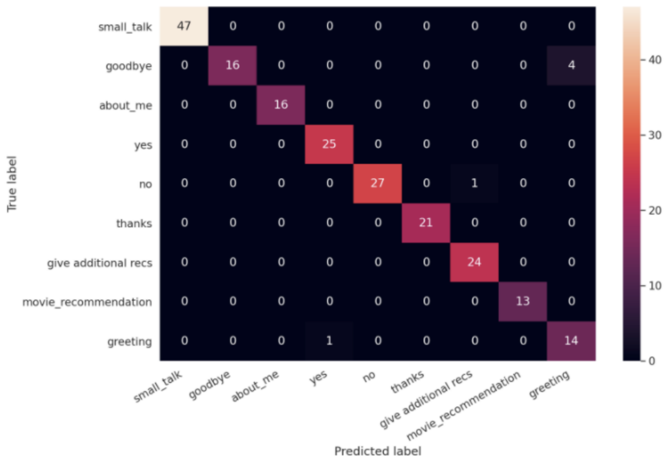


Figure 5: Classification Heat Map

for misclassification is perhaps because of the word “these”, which are commonly included in some of the phrases that are in the training data. Example phrases under the “give additional recs” class include “I've already seen these”, “I am not feeling these movies”, and “I don't like these movies”. Based on the high similarity between “I'll be sure to check these out”, it would've wrongly labeled it as wanting more recommendations.

Conclusion/Next Steps

The large accuracy difference in the first chatbot iteration involving DNN and the BERT approach signifies BERT's efficacy in intent recognition that helps

	precision	recall	f1-score	support
small_talk	1.00	1.00	1.00	47
goodbye	1.00	0.80	0.89	20
about_me	1.00	1.00	1.00	16
yes	0.96	1.00	0.98	25
no	1.00	0.96	0.98	28
thanks	1.00	1.00	1.00	21
give additional recs	0.96	1.00	0.98	24
movie_recommendation	1.00	1.00	1.00	13
greeting	0.78	0.93	0.85	15
accuracy			0.97	209
macro avg	0.97	0.97	0.96	209
weighted avg	0.97	0.97	0.97	209

Figure 6: Class Results

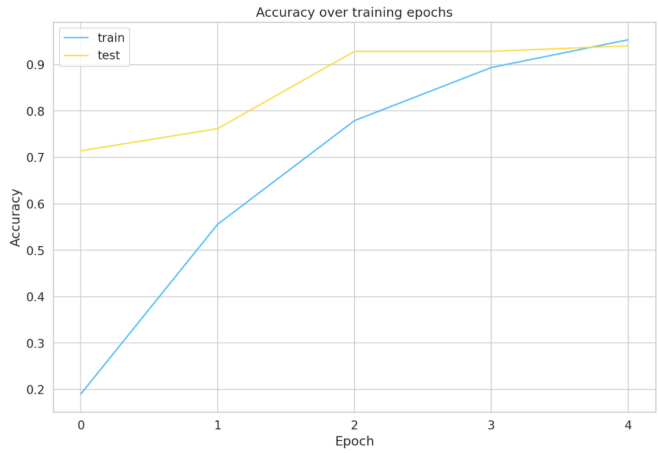


Figure 7: Accuracy Over Epochs

to facilitate a fairly free flowing conversation between the user and chatbot. BERT achieved consistently high precision, recall, and F1 scores across all intent classes, with the lowest recall score being 80 percent, belonging to the “goodbye” intent class.

In future iterations of our recommendation chatbot, we would hope to explore other architectures like T5. We would have also liked to incorporate BERT in not just a chatbot intent classification but also in the movie recommendation aspect itself. This paper [8] explains the usage of BERT4Rec, a sequential recommendation model that addresses the limitations of sequential neural network models in that they are unidirectional. The proposed model is bidirectional and is said to outperform various sequential models. Another BERT variation we were considering was TOD-BERT [9], specifically for its strengths in task-oriented dialogue. The paper argues that pre-trained language models are not very effective, and in the results section, TOD-BERT outperforms BERT itself in tasks like intent recognition and response selection. In our project, the chatbot response selection is very straightforward; there are pre-generated responses to certain intents, and it randomly selects one to generate to the user, but having a more nuanced approach to response selection would have improved the conversation.

Additionally, BERT BASE specifically was used for our chatbot, but BERT LARGE could be incorporated in a future iteration. We would also consider strengthening the intent dataset and test the efficiency of BERT on many different dataset sizes. Based on this paper about using BERT for more NLP tasks and movie recommendations, we were also heavily inspired by this paper [10] to try and expand into looking at BERT's functionality even further, and observe how much inherent knowledge that BERT pre-trained models possess about films.

We find that the advantage of creating our BERT recommendation chatot is its generalizable nature and universality. Other contexts this could apply to are restaurant recommendations, vacation destination suggestions, and other possibilities in various realms. The main changes would be altering the intent files and the dataset to make it more context specific, but otherwise, the functions and code that trigger the BERT intent classification and recommendation would be very similar. Our goal for this project was to create not only a movie recommender system, but also a personable chat bot that can converse with its user. We would hope to better our approach in future iterations.

References

[1] Xiao, Ziang, et al. "Tell Me About Yourself: Using an AI-Powered Chatbot to Conduct Conversational Surveys with Open-Ended Questions.". Jan 2020. arXiv.org, https://arxiv.org/pdf/1905.10700

[2] Guo, Yuchen, et al. "Topic-Aware Chatbot Using Recurrent Neural Networks and Nonnegative Matrix Factorization." ArXiv:1912.00315 [Cs, Math, Stat], Dec. 2019. arXiv.org, http://arxiv.org/abs/1912.00315

[3] gk. "Contextual Chatbots with Tensorflow." Medium, Chatbots Magazine, 9 Sept. 2017, chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077.

[4] Nutan. "Step by Step Intent Recognition With BERT." Medium, Medium, 18 Oct. 2020, medium.com/@nutanbhogendrasharma/step-by-step-intent-recognition-with-bert-1473202b8597.

[5] Google-Research. "Google-Research/Bert." GitHub, 11 Mar. 2020, github.com/google-research/bert/blob/master/README.md.

[6] Elkhattam, Ahmed. "IMDb Movie Recommendation Chatbot." Medium, Medium, 7 Dec. 2020, a-elkhattam.medium.com/imdb-movie-recommendation-chatbot-942f84dfa0dc.

[7] Li, Bai, et al. "Real-World Conversational AI for Hotel Bookings." ArXiv:1908.10001 [Cs, Stat], Aug. 2019. arXiv.org, http://arxiv.org/abs/1908.10001.

[8] Sun, Fei, et al. "BERT4Rec." Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, doi:10.1145/3357384.3357895.

[9] Wu, Chien-Sheng, et al. "TOD-BERT: Pre-Trained Natural Language Understanding for Task-Oriented Dialogue." Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, doi:10.18653/v1/2020.emnlp-main.66.

[10] Penha, Gustavo, and Claudia Hauff. "What Does BERT Know about Books, Movies and Music? Probing BERT for Conversational Recommendation." Fourteenth ACM Conference on Recommender Systems, 2020, doi:10.1145/3383313.3412249.