Internship Project Draft

The Relevance feedback mechanism required us to filter the recommendations by analysing the conversation. The Speech was converted to text and then the conversation was analysed sentence by sentence.

Analysing everything conversation is rather challenging. We identified three types of keywords that needed to be extracted from the conversation namely genre, movie and actor along with the sentiment attached to these keywords. An algorithm was then developed to update the recommendations based on the Keyword – Sentiment pair.

**Keyword Extraction**:

1) Dataset: We did not find any existing dataset relevant for our purpose and hence we had to crawl sentences from movie forums and manually annotate them based on our needs. We collected nearly 500 sentences for this purpose.

2) Genre Extraction: We found from the dataset that there are an exhaustive number of ways in which one can refer to a genre and hence for the purpose of genre extraction, we simply searched through the text for presence of any genre.

3) Movie Extraction: The List of Movies is not as exhaustive as genre and movie names usually have more than one words. Movie Extraction was done in two steps: movie tagging and movie matching.

* Movie Tagging: The Sentences were parsed to detect the presence of movie name. This was done in three ways.

1. Naïve Method: Get all the Noun Chunks in the Sentence and mark them all as movie names. This was too naïve and lots of false words were detected.
2. Rule Based Method: Carve Rules on what chunks might possibly be movie name in the conversation. The Rule based method actually fared pretty well and it was able to tag a large percentage of movies. However, the precision was still low and lots of other chunks and star names were also detected as movies.
3. Model Based Approach: Tag the words in the sentences as ‘I’, ‘O’, and ‘B’. Make a classifier model by training a gradient boosting machine. Some of the features taken for the purpose of training this classifier was:
   1. Tags of previous and current words in the sentence
   2. Whether the word is starting or ending word of the sentence
   3. Vector representation of words given by a word2vec model trained on a huge movie review dataset of 100000 reviews.
   4. Frequency of Occurrence of Words in the movie Vocabulary built by those 100000 movie reviews.
   5. Presence of an adjective or presence of a synonym of watch or other intentional word in the neighbourhood.

Although the performance of this classifier was good, A few words were missed out in the middle of the chunk as the classifier did not care about the sequence. To take into account the sequence, a hidden markov model was trained with three states ‘I’,’O’ and ‘B’. Since the dataset was not large enough, individual words could not be taken as observations and hence the POS tags were taken as observations. A bigram model was created to compute the likelihood of POS tags occurring in each states and the transition probabilities were learnt from the data. Although the performance of this HMM Model was not as good as the performance of the classifier, the transition probabilities learnt helped to improve the gradient boosted machine’s performance. Wherever the transition probability for one state to another was too low and even then the classifier predicted that state, it was detected as an anomaly and HMM tag was used in that place. This actually improved the recall of the machine by more than 5% although it brought down accuracy marginally.

Combination Algorithm:

for sentence in dataset:

for words in sentence:

if ( transition\_probability (prev\_word\_tag,current\_word\_tag) < threshold)

current\_word\_tag = current\_hmm\_word\_tag

else if ( emission\_probability(current\_word\_tag,current\_word\_postag) < threshold)

current\_word\_tag = current\_hmm\_word\_tag

* Movie Matching: Even when the parts of sentences were classified into movie and non-movie chunks, the real movie which was referred to was yet to be identified. To make the identification robust, the movie needed to be detected even if There were slight mistakes in google text to speech conversion or only a part of movie and not the full name of movie was spoken.

For this purpose, an algorithm based Levenshtein distance was developed to return matching results which were then filtered on the basis of the context of the discussion.

The movie names which had the least Levenshtein distance to the movie detected were selected, If the distances were close enough, more than one movies were picked form the lot and those movies were then score again on the basis of context.

movie\_score = Levenshtien\_Distance (detected name, actual name)

+ alpha \*( movie\_genres \* genres\_referred\_recently)

+ beta \* ( movie\_stars \* star\_refereed\_recently)

The Movie matching algorithm was actually very robust and gave a precision of 87% and recall of 96% if all the movies were correctly tagged.

However, Since the precision and recall of the tagging were lower, the final Extraction precision was around 76% and recall was around 78%. These results are expected to better for the conversation since the conversation is supposed to have less frequent context change as compared to the discussions in the forums.

4) Stars Extraction: Similar to movies, directors and actors list is also very huge. We had about 15000 actors in our database. This was however a simpler task as compared to the movie extraction. Similar approach was followed. Tagging was done first followed by matching.

* Stars Tagging: Sentences were tagged to detect the presence of a person’s Name. Three approaches were tried.

1. Naïve Method: Get all the Noun Chunks which have a low count in the vocabulary built by movie corpus and those which don’t exist in dictionary. This method had a good recall but precision was low as movie names were also tagged along.
2. Stanford Tagger: For the purpose of person tagging, we used Stanford NER Tagger. It performed really well. However, late we found out that it did not work well if the names were not capitalised. And hence the recall was not up to our requirements.
3. Model Based Approach: For our purpose, we needed to create a model which even if did not have a precision nearly as good as Stanford Tagger but could work in presence of unreliable capitalization. A gradient boosted classification model was hence trained. These features were used:
   1. Tags of the Previous and the Current Words
   2. Whether the word is starting or the end of the sentence
   3. Vector representation of words given by a word2vec model trained on a huge movie review dataset of 100000 reviews.
   4. Tags of the Stanford NER Tagger.

The Sequence did not matter that much in Stars Extraction and the model already had fairly high precision and recall and hence it was used as the final model in our system.

For the purpose of matching, the same algorithm was used. The Formula used to calculate the score was:

actor\_score = Levenshtien\_Distance (detected name, actual name)

+ alpha \*( actor\_genres \* genres\_referred\_recently)

+ beta \* ( actor\_movies \* movies\_refereed\_recently)

Final Precision and Recall in Stars Extraction was better than movie Extraction. Precision and recall were both around 90%.

4) The Determiner’s Problem: Another challenge in the conversation analysis was posed by the determiners. Often when people are continuing the conversation, they use determiners like it, this that instead of actually taking a movie name. This Problem could have a better solution, but we stuck to the naïve approach and assigned these determiners to the keyword extracted earlier. A set of rules were made for this purpose like third person pronouns were assumed to be referring to an actor or director talked about last. Similarly, plural determiners like “these” attached reference to past three keywords rather than one, and the things like “these movies” attached reference to the genre attached. Only 5 rules were made none so that the rules do not over fit our observations.

**Sentiment for Keyword:**

After finding the keyword, the next problem was to attach it to its sentiment. The simplest idea was to attach the sentiment of the sentence in which the keyword was found to the keyword. For the purpose of the sentiment classification, nltk sentiment api was to be used. However, this idea had certain problems.

1. Classifiers like NLTK have been trained on large documents and hence are not that much reliable for sentence level sentiment analysis.
2. Positive Intent like I want to watch something was classified as neutral instead of being classified as positive.
3. The movie or genre names like horror often altered the sentiment of the sentence.
4. Sentences in form of questions were wrongly classified. Why don’t we watch Inception gave a wrong sentiment?
5. In conversation people often use conjunctions which mean that different parts of sentences might carry contrasting sentiments for keywords and also in conversation people frequently use comparisons like I like A more than B.

The Solution was to train a classifier for our purpose using a simple forest based algorithm with features that could also classify positive intent as positive sentiment. Since we had already extracted the keywords, to make sure that the keywords did not alter the sentiment was simple enough. We replaced the keywords with a neutral word. This actually did the job and sentences like “ I think We should watch wrong turn” did not give negative sentiments.

The same 500 Sentence Dataset was annotated. 60% of the sentences were used for training the model. The following features were used in the classifier:

1. Positive, Negative and Neutral Values and the label of the NLTK Sentiment Analyser.
2. Subjectivity and Polarity Values of the TextBlob Sentiment Classifier.
3. Presence of Words showing positive or Negative Intention in the sentence.
4. Odd or Even Count of Negation Words like not or never in the sentence.
5. Presence of a “WP” tag in the sentence, which essentially means presence of a question word in the sentence.
6. Count of Positive Sentiment Words in the sentence.
7. Count of Negative Sentiment Words in the sentence.

The repeated cross validation technique showed the accuracy to be about 79%. Although it was not as good as we wanted. For our purpose, it was certainly an improvement over what we already had.

This new model more or less solved the first 4 problems. However, the last problem i.e. the conjunction and comparison in sentence remained. Initially, we tried to come up with some model that could be trained on parse tree and subtree but that actually made the problem complicated and also took a lot of time and hence we decided to use the rule based approach based on linguistic analysis. conjunctions are too many and are used in different ways. We obviously could not draft rules for all of them and so 8 most commonly used conjunctions that could possibly be used to convey contrasting sentiments for keywords were picked and a total of 16 rules were made. The idea was to either treat sentences differently in presence of conjunction by removing the conjunction or attaching the sentiment of a keyword to the other part of the sentence by adjusting the polarity. Stanford Parser was used to parse the sentence and the rules were made for the parse tree. Rules for Conjunction have been mentioned in Appendix 1.

Similarly, a few rules were drafted for comparative sentences as well, which have been mentioned in Appendix 2.

The output of the Conversation Analyser was the Keyword along with the sentiment for each user in the session. A few tags were also associated with the Keyword like whether the keyword was actually referred or inferred from the determiner which was later used for user-user agreement graph and a tag which determined whether the movie mentioned was referred in the past tense. The accuracy of this tag was not up to standards and hence its use was scraped.

Appendix 1: (Conjunctive Rules)

Appendix 2: (Comparative Rules)

Appendix 3: (Movie Chunk Rules)