Final Project: Retrieval Model for a Code-Mixed Query

What is the task, and why is it important to users?

Task:
Sentiment Classification on a Code Mixed dataset
The task of the project is to classify a multilingual query or a document into three sentiments- positive, negative, or neutral. Since there is a scarcity of code mixed data, there is limited work done on the same.

Importance:
According to a study done in 2021, there are approximately 43% of the population is bilingual. A further 17% of people are multilingual. More than half of the world's population can speak more than one language. Another trend seen in the multilingual speaking area is that the people in the age group of 16-24 years are more likely to speak more than one language. This is the same age group that is seen to be the most active on social media platforms like Twitter, Instagram, and Facebook. Hence there has been a rise in the usage of code mixed sentences, increasing the data available in this area making it more necessary for NLP engineers to work on this problem.
Although information retrieval systems and natural language processing systems have seemed to have gained high accuracy in the cross-lingual text classification area, the code mixed text datasets themselves seem to be very scarce. 30% of queries to Ruuh (a Microsoft chatbot) are code mixed. Alexa gets code-mixed speech inputs. Around 2-30% of posts on Twitter and Facebook are code mixed. Observing this shift in the trend of linguistics, having a multilingual retrieval model seems to be a necessity more than an added feature to a search engine. A study of 830k tweets from the Hindi-English bilinguals also indicated that the native language, Hindi was more preferred for expressing negative sentiments while English was preferred for the positive ones. These statistics also tell us that multilingual retrieval models are not only restricted to understanding the meaning of the query, but also the sentiment.

In general, what do queries look like?

A code mixed query would look something like the following:
(The following texts have been manually taken from Twitter based on various topics)

1. "Bapuji after every drama: 'just chill chill, just chill' Bapuji is a legend." #Anupama
2. "The very best part of Anupama; her aankhon ke taare in one frame."

3. "Chirag aur Saisha ka rishta kya kehlata hai? I am literally confused. Ye kya lagte hain ek dusre ke?"
4. "Had a meeting with a delegation from Panchayat Sangiote of Mendhar Constituency."

The first three tweets are about a Hindi TV show. Only those tweets have been picked up which could potentially be used in a search engine like Google to find out more information about the topic. These could also be an input to the search engine of the OTT platform streaming the shows. The sentiment and the context both would decide what the results of the search would look like. The first tweet mostly is written in English, but the main subject of the sentence is 'Babuji', which means father in Hindi. The context of this sentence revolves around 'Babuji' being in a positive light. If we take a look at the last example, it is about a politician informing us on Twitter about a meeting he/she had. The sentiment of this statement solely could be neutral and hence giving us relevant results. We would also have to understand the meaning of words like 'Panchayat' for getting accurate results.

What kinds of results would be relevant to these queries? How many relevant results should there be per query?

Given that the queries were entered in a search engine like Google, we would expect to see results that were relevant to the query in terms of its context and have the same sentiment.
Each of the samples queries given above

1. For the first query, the expected result would be documents related to the TV show 'Anupama'. We would expect that the resulted documents have information in the context of 'Babuji' who is the main subject of the query. The subject of the query has been shown in a good light and hence a document containing positive information about the subject would be the most relevant one. A naive way would be just matching the keywords of the query to the documents in the collection which would not guarantee that the resulting document is relevant.
2. For the second query, we again need documents relevant to the TV show Anupama. The key phrase here is, 'aankhon ke taare', which translates to 'the stars of someone's eyes' but means 'the apple of one's eye'. We would require the search engine to either translate and understand the context of such Hindi phrases or just map these Hindi phrases directly to the context in English. Here, the search engine can understand that the query is positive using the word 'best' but it would be much clearer if the system could take into consideration the Hindi

part of the sentence. A relevant result to this would contain documents marked as positive and in a similar context to the query.

3. The third query also talks about a Hindi TV show called, 'ye rishta kya kehlata hai'. The tricky part of this query is that this phrase directly translates to 'what is this relationship called' which is used as a verb in the sentence and not as a noun. The search engine would have to figure out whether the user is talking about the TV show or is asking about how the two people are related (who could be any two people from outside the show). In this case, the user is talking about the show itself in a confused sense. Since we are only classifying the emotions in 3 categories: positive, negative and neutral, this query would come under the neutral category. Hence relevant documents would contain information about the two characters of the TV show mentioned, and how they are related in a neutral sense.
4. The last tweet is about a meeting held by a politician/leader. The word 'Panchayat' is the keyword here, meaning a village council. 'Sangiote' is a region in India. The sentiment of this query would be neutral since it just informs about a meeting held in a particular village council. The expected result of this query should be documents having the information of the recent meetings held in the village council of Sangiote.

Given a few examples, it is clear that we would filter the results with various things before giving the results. The most basic way search engines operate is by spitting out results (documents) containing the maximum amount of keywords, which works in most cases but not always. In our system, we would check the usage of maximum keywords, then we would have the model understand the context the user has written the query which might not be clear from the query but could be extracted from the previous searches of the user. The last step would be finding documents of a similar sentimental tone. Given all these filters, the number of relevant results in many cases can be expected to be lesser than usual. Hence a maximum of 3 close to accurate results would be enough for one query.

How should the results be organized (ranked list, clusters, summaries, etc.)?

The best way of representing the results is using a ranked list of the documents. Before that, we would have a ranked list of documents without considering the sentiment of the document and the query. We would then cluster the documents based on their respective sentiments: neutral, positive or negative. Next, we would again rank the documents and give more weightage to the documents with the sentiment same as that of the query. Hence, the documents with the same sentiment may be ranked higher than the ones with different sentiments. We would manually have to check for every query if the results given are relevant.

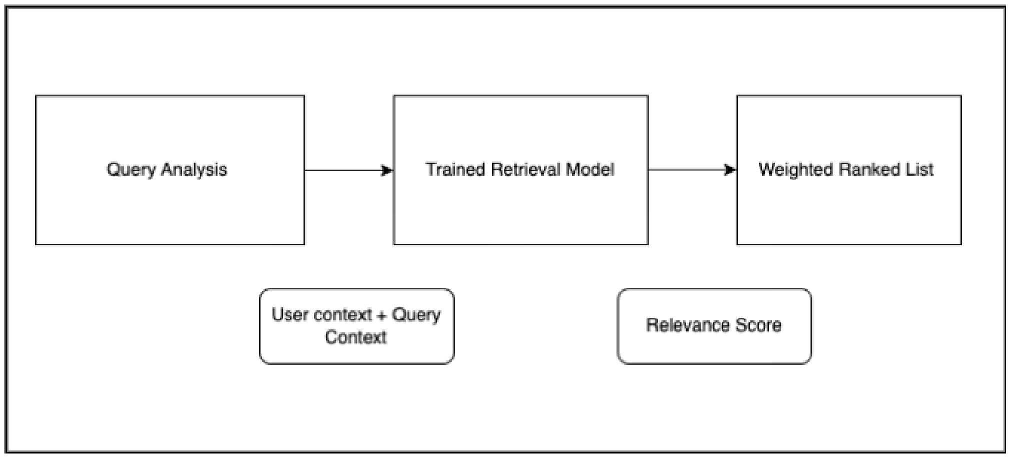
What evaluation metrics would be appropriate for this task?

The evaluation metric appropriate for this task is the Mean Average Precision for the retrieval model. This is used because it gives more importance to the precision of the documents that are ranked at the top. This is required in this model since the latter results tend to get more irrelevant and hence we need to focus on the first few results.

What modelling approach(es) would you propose for solving this task? What software architectural components would need to be built? What training data would need to be collected and/or annotated?

As mentioned earlier, the retrieval system would work just like the ones existing now taking into consideration the search history of the user, the location and other such factors while retrieving the ranked list of documents for a query. The new features that we have added in this project are - a code mixed query would be able to return appropriate results in both languages and that the sentiment of the query and the document are taken into consideration as well while getting a ranked list. The code mixed query would be classified into three sentiments, neutral, positive and negative. The same would be done for all the documents present in the corpus. While ranking the documents with the same sentiment would be given more weightage making them appear on top of the ranked list.
There are three main components to this retrieval system:

1. Analysis of the code - mixed query for retrieving the sentiment and getting other features
2. The retrieval model will take the code mixed query as the input and will try to find documents matching the query the most based on various factors, like the keywords matching, the user's search history, location and also the sentiment. This specific model needs to be trained on a code mixed data for classifying the sentiments
3. The ranked list of documents is based on the weighted relevance score where we first get the ranked list of documents with the simple model and then get a ranked list after applying sentiment classification on the code mixed data. Then we would manually have to check if the results we get after this process give us more relevant documents.



I would be discussing the three points in detail here:

Analysis of the query:
The query was analysed based on the information given in the query. Phrases, context and references from topics like TV shows, politics, current affairs, etc are the ones that are given more importance while understanding what the documents retrieved should be ranked on. The query is also code mixed, hence the model should be trained on the languages that are going to be code-mixed or there should be an interpreter of languages. The drawback, in this case, is that the model is not trained on the languages that are being input and hence these parts might go uninterpreted which defeats the purpose of the model. We then classify the query as positive, negative or neutral.

Retrieval model:
Once we have the query input, we need to give it to the retrieval model. The regular retrieval models like Google, use features like keyword matching, context from the search history of the user, location and other such factors playing a part in the getting a list of ranked documents. We use Manhattan distance to find the similarity between the text in the query and the documents. Word embeddings are also used to know the context and get the document of a similar context.
In this project, we would be using these factors too, and use a model trained on Hindi-English code-mixed data. We also use sentiment classification of the documents as an added feature for ranking the documents.

Sentiment Analysis on code mixed data component:

Data cleaning is one of the most important steps in the problem. If we want to classify code-mixed text only we should filter out the non-code-mixed text before training the model which has shown to improve the F1 score significantly by 50%. If we choose to have a monolingual model as well, then separating the monolingual languages in their respective places and then training the model accordingly would give good results. Normalization is also an important step, which is done using various basic features like words, lemmas, POS tagging, and Levenshtein distance in the above papers along with syntactic features and N-grams for word and feature dependencies. Class imbalance is an important factor considered which is solved by resampling techniques like oversampling and undersampling which increases the F1 score by 2%. Another observation is that the less accuracy got could be attributed to the less data we have.
Logistic regression is the state-of-art model for this data. Another model used is the one that consists of the LSTM attention-based model giving comparable performance. The F1 score and accuracy are the evaluation metrics that are chosen for this problem. The future scope of research in this area could be done by experimenting with the data on deep learning models like the CNN architecture due to its ability to exploit spatially-local correlations and it is seen that code-switched sentences tend to contain continuous slices of text written in the same language. We could also try to work on the challenges mentioned in the introduction that mainly deal with the data cleaning and normalizing. We could also try to work on this problem with unsupervised learning since there seems to be a scarcity of labelled data.

Using all these components, we train the model to retrieve the documents that are the most similar based on the above-mentioned factors using the weighted relevance score.

Ranked list:
The ranked list is the final output of the retrieval model after it ranks the documents based on the variety of factors mentioned above. The documents on the top are expected to be the most relevant to the query in terms of the information, keywords, context, and the sentiment that the user is trying to search for. As we move down the list, the relevancy of these might go on reducing. The last few results are even expected to be irrelevant and ranked randomly. These rankings would be done based on the weighted relevancy score that we obtain from the retrieval model.

Data Collection, Annotation:
The data - documents were collected from the social media app, Twitter. The data collected is Hindi-English code-mixed based on TV shows and politics. The data was also annotated by me based on the context and the sentiment of the information in the document.

The second section should consist of sample queries, narratives, and relevance judgments. After formulating a query, you should write a narrative describing what the user is trying to accomplish with that query and what criteria the user might use for judging results to be relevant or non-relevant.

For the sample queries, we use the same queries given in the examples above.

1. "Bapuji after every drama: 'just chill chill, just chill' Bapuji is a legend." #Anupama
2. "Chirag aur Saisha ka rishta kya kehlata hai? I am literally confused. Ye kya lagte hain ek dusre ke?"
3. "Had a meeting with a delegation from Panchayat Sangiote of Mendhar Constituency."

10 Hindi-English code-mixed documents are in the corpus that we need to rank based on the query we get in the search engine. Some of the documents are relevant to the queries given above while the other are irrelevant. The ten documents are uploaded here:
https://github.com/ketakiii/Sentiment-Classification-in-a-Code-Mixed-Environment/tree/main/Documents

Relevance Judgement and Narratives

- "Bapuji after every drama: 'just chill chill, just chill' Bapuji is a legend." #Anupama

As we discussed above, the meaning of 'Bapuji' is 'father'. The tone of the query is positive and also mentions the keyword Anupama which is a Hindi TV show. Hence, we would expect a document containing information about the TV show Anupama with the context of Bapuji shown in a positive light. The only challenge is that the search engine needs to understand this query regarding a TV show named Anupama.
Documents 1 through document 3 have information concerning the TV show Anupama. Although, just the first two documents should be classified as positive. Document 5 is not about the TV show Anupama but contains a reference to 'Baap' which is the root word for 'Baapu'. But the sentiment of the document is negative. Documents 7, 8 and 10 are positive but not about the TV show or anything with that reference.
Then, we would rank the documents without taking into consideration the sentiment weightage. Documents 1 through 3 would rank the highest, then document 5 would come up and then the rest of the documents would be ranked. Then we weigh the sentiment and rank the documents again according to the sentiment of the query. Hence, document 1 or document 2 would be the top ones to be ranked. Then document

5 would be the third one to be ranked. Document 3 would be ranked next despite being negative, but it contains the information about the TV show Anupama. Then, the rest of the documents would be ranked in some random order.

Ranking:
Document 2 : 11.35
Document 1 : 10.91
Document 3: 8.99
Document 5: 4.51
Document 7: 3.33
Document 8: 2.0
Document 10: 1.79
Document 9: 1.34

- "Chirag aur Saisha ka rishta kya kehlata hai? I am literally confused. Ye kya lagte hain ek dusre ke?"

The query talks about the TV show 'ye rishta kya kehlata hai?'. The search engine needs to first understand that the user is talking about this TV show although it means what the relation of the two people means literally. Hence, based on the user history and other features it should understand that it is about a TV show. The sentiment is neutral and hence it would expect documents related to this TV show with a neutral sentiment.
Document 4 contains information about that particular TV show. Without the sentiment classification, when we rank the documents, document 5 would be ranked on top based on the context. When we rerank the documents also adding in the weights of the sentiments document 5 would rank on top as it's neutral and hence, this document should rank at the top of the list. The rest of the documents are unrelated to this query. Document 9 is also classified as neutral hence might come up in the ranking. The rest of the documents would be ranked in some random order since they are unrelated.

Ranking:
Document 5 : 11.35
Document 9 : 10.91
Document 3: 8.99
Document 1: 4.51
Document 7: 3.33
Document 8: 2.0
Document 10: 1.79

Document 2: 1.34

- "Had a meeting with a delegation from Panchayat Sangiote of Mendhar Constituency."

The query talks about a meeting with a government constitution. The word 'Panchayat' means a village council. The context of this query is about either an official or a politician having a meeting. Hence, even if we don't get the keyword matching, in this case, word embeddings would help us understand which context the query. The vectors for this query and a document containing any information about governance/politics would be close. Hence, those documents would be ranked higher.
The sentiment of the query is neutral hence the expected document would also be of a neutral tone.
Here, document 6 talks about municipalities, police, etc which lies closer to the query vector but has a negative sentiment to it. Document 7 talks about American politics whose vector might lie farther away from that of the query since it talks about a village council in Mendhar which is a village in India. It has a positive sentiment. Document 8 contains a tweet from an official coming into office of the Panchayat with a positive sentiment. And the last document 10, contains the words Panchayat but in a different context. Panchayat is a TV show and the document talks about being excited for season 2 of the same, with a positive sentiment. The rest of the documents are irrelevant in terms of the content but document 4 has a neutral sentiment. The ranking of the documents might be as follows: Either document 6 or 8 ranked at the top, followed by document 7, then document 10 and then the rest due to their irrelevant content.

Ranking:
Document 6 : 11.35
Document 8 : 10.91
Document 7: 8.99
Document 10: 4.51
Document 1: 3.33
Document 3: 2.0
Document 4: 1.79
Document 2: 1.34

References:

- **Robust Deep Learning Based Sentiment Classification of Code-Mixed Text:** Siddhartha Mukherjee, Vinuthkumar Prasan, Anish Nediyanchath, Manan Shah, Nikhil Kumar Samsung R&D Institute India, Bangalore
- **Sentimental analysis from imbalanced code‑mixed data using machine learning approaches**: R. Srinivasan1 · C. N. Subalalitha1
- **Sentiment Analysis of Code-Mixed Text: A Review:** Nurul Husna Mahadzir1, Mohd Faizal Omar2*, Mohd Nasrun Mohd Nawi3, Anas A. Salameh4, Kasmaruddin Che Hussin5
- https://www.youtube.com/watch?v=YWr2pV_hFYs&list=PL7DyM0qTpj_IGVCrTQvLA89SrS_2gaTZ4&index=4&t=2273s