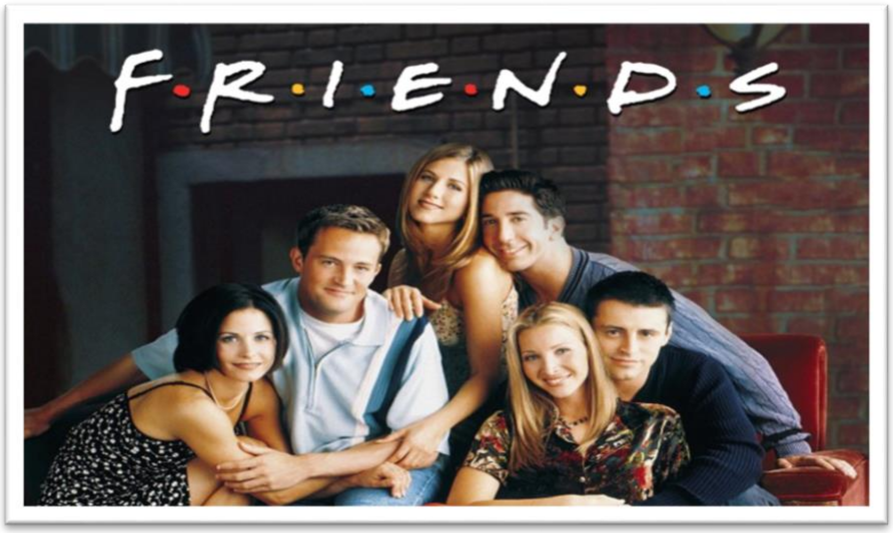
Introduction

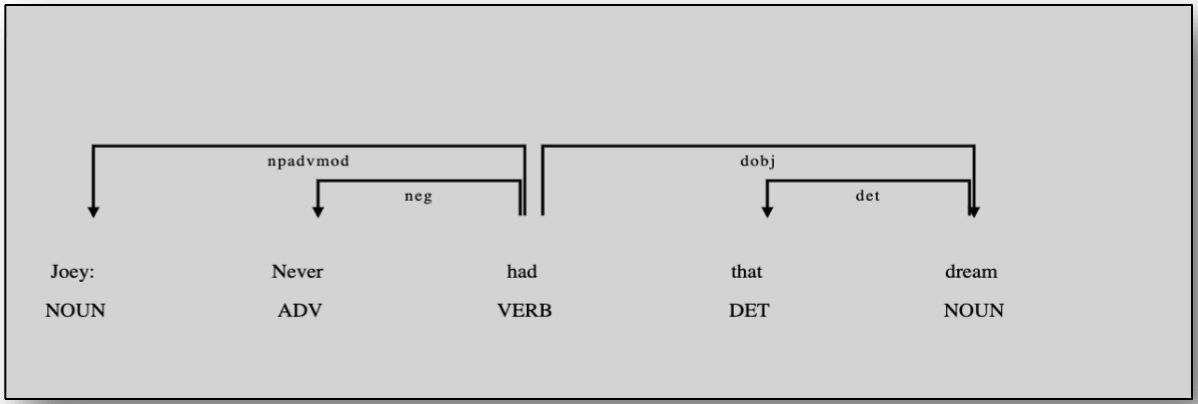


I have used subtitles from 'Friends' series from https://fangj.github.io/friends/. I parsed multiple webpages of the mentioned link and have extracted text of Season 1 from episode 1-12 using Python's request library and Beautiful Soup to make the html readable. I also handled failed webpage retrievals by printing an error message with the status code. I then compiled the text into a string and list of paragraphs, wherein the total number of tokens was 57,794 (words and punctuation). The summary features dialogue from these episodes, illustrating human interaction and story development. "Friends," the beloved TV series, portrays the lives of five New York friends. The conversation is characterized by cultural references, humor and friendliness.

I used this data for analysis for multiple reasons. First, "Friends" is an iconic show with a large fan base, making it an interesting topic for textual analysis. Analysis of the dialogue can reveal linguistic patterns, character development, and the nature of humor and dialogue in the show. I was eager to study the relationship pattern between different characters, analyze their moods/sentiments throughout the show, identify frequent topic discussed among them majority times & find ways to get an overall summary of the show. Also, by examining the sentiments and recurring themes, some insights could be shed into the cultural and societal values reflected in the show. Furthermore, this research can show how the discourse of the show reflects the cultural climate of the 1990s and early 2000s. By examining the frequency and context of specific words and phrases, my goal is to identify trends and themes that contribute to the show's lasting popularity. This study could be insightful for those interested in the show as well as for researchers focused on media studies, linguistics, and cultural analysis.

Processing Text

Several interesting findings came from analyzing the raw content of the "Friends" subtitles. Semi-speech analysis showed that the conversation was dominated by ‘Nouns’ and ‘Verbs’, which are characteristic of conversational content. People often use personal pronouns, underscoring the show’s focus on interpersonal relationships and everyday interactions. Adjectives were few, but when used, they added significant detail to the characters and humor in the dialogue.



The above image is using a text from the series “Joey never had that dream”. I have implemented POS tag on this text after loading the spacy model. Here "Joey" is a proper noun referring to a specific character from the Friends series. Sentence above contains a mix of different parts of speech, with nouns being prominent. In this short sentence, two out of five words are nouns (Joey and dream), highlighting the importance of nouns in conveying the key entities or subjects within the text. This analysis demonstrates the significance of nouns in representing key elements within a sentence and how POS tagging can be used to enhance our understanding of textual data.

Still after proper tagging of words in few texts, I have encountered some difficulties with proper punctuation because the language in "Friends" tends to be more informal and conversational. The show is full of colloquialisms and slang, which complicated the part of speech (POS) tagging process. For example, informal language and the unique ways different characters speak lead to incorrect classification, as traditional POS marking systems cannot adequately handle this variation which eventually lead to errors where words are tagged incorrectly or their grammatical roles misinterpreted, making it difficult to analyze the text accurately.

For example, in the text below:

***“Could you be any more annoying*?”**

- ‘*be*’ = verb, but it's part of a common expression that might confuse the system.
- ‘*annoying*’ = adjective, but in casual speech its role can be a bit unclear.

Because this sentence uses informal and idiomatic language, a POS tagging system might have trouble correctly identifying the parts of speech.

To improve the evaluation, it would be useful to include a more robust POS tagging model further trained on conversations and informal information. Since, Friends is a type of series which has a lot of casual talks & slangs, so it would be better to have a model that understands these kinds of dialogues which can eventually increase accuracy and make the tagging look meaningful. Additionally, fine-tuning the existing model in a subset of the "Friends" script can help reduce errors.

Important generalization/normalization steps which one should follow include converting all text to lowercase to avoid treating same word in different cases as separate entities, removing punctuation marks, and inserting words into references to avoid unnecessary noise and focus on the context of the text, another key step should be lemmatization which involves reducing words to their basic/root form (e.g. 'running' to 'run') rather than stemming, which might eventually give less meaningful roots. These techniques contribute to the consistency of the text, making it easier to identify common patterns and themes.

A surprising finding was the frequency with which some words and phrases are there, such as "you know" and "I mean", which are commonly used in spoken language but rarely in written language.

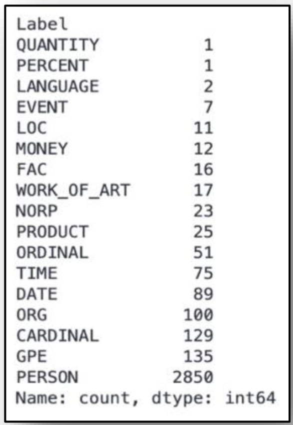
For example:

"I mean, I just don't think that's a good idea, you know?"

This is just one subtitle but there was majority of instances like this in the complete subtitles text. These filler words makes the dialogues sound more natural and relatable but they can affect how often words are counted if not handled carefully. This shows just how important it is to understand the context in which these words are used. This correctly emphasizes the importance of context in understanding human interaction and the story as a whole.

Information Extraction

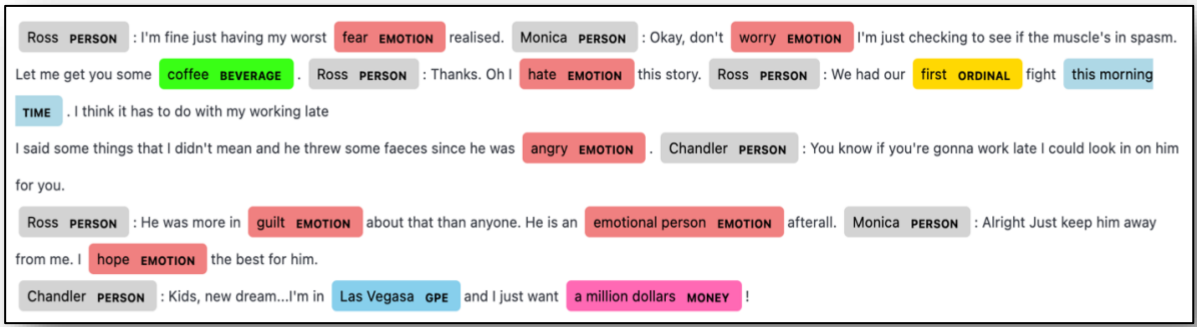
Using the SpaCy library, I extracted named objects from the subtitle data, including groups such as PERSON, ORG (organization) , GPE (geopolitical entities), EVENT, LOC (locations) and LAW. I tokenized the subtitles and identified entities with the text. The extracted features were sorted in a Data Frame, and their frequencies were calculated and analyzed. The results showed a range of features, with PERSON and GPE being the most frequent, reflecting the conversational nature of data from the "Friends" TV shows. Notable companies included people's names such as "Ross", "Rachel", "Monica". These entities appeared frequently, reflecting their central roles in the dialogue. Locations like "New York" was also prominent, as the setting of the show is based in New York City.



```
Label
QUANTITY          1
PERCENT           1
LANGUAGE          2
EVENT             7
LOC              11
MONEY            12
FAC              16
WORK_OF_ART      17
NORP             23
PRODUCT          25
ORDINAL          51
TIME             75
DATE             89
ORG             100
CARDINAL        129
GPE             135
PERSON         2850
Name: count, dtype: int64
```

The entity extraction also provided insights into recurring themes and settings within the series. By analyzing the frequency and context of these entities, I was able to identify key plot points, character relationships, and significant locations.

I also added new entity types called "**EMOTION**" and "**BEVERAGE**" using SpaCy's entity ruler. I defined specific patterns that match words related to emotions like fear, worry, hate, etc., and also a pattern for the word "coffee" as a beverage. These patterns help SpaCy recognize and label these entities in text. After defining these patterns, I processed a sample text from my preprocessed dataset. SpaCy analyzed the text based on these rules and highlighted entities such as emotions (e.g., fear, hate) and beverages (e.g., coffee) using different colors. This visualization helps to easily identify and understand where these entities appear in the text.



Ross PERSON : I'm fine just having my worst fear EMOTION realised. Monica PERSON : Okay, don't worry EMOTION I'm just checking to see if the muscle's in spasm. Let me get you some coffee BEVERAGE . Ross PERSON : Thanks. Oh I hate EMOTION this story. Ross PERSON : We had our first ORDINAL fight this morning TIME . I think it has to do with my working late
I said some things that I didn't mean and he threw some faeces since he was angry EMOTION . Chandler PERSON : You know if you're gonna work late I could look in on him for you.
Ross PERSON : He was more in guilt EMOTION about that than anyone. He is an emotional person EMOTION afterall. Monica PERSON : Alright Just keep him away from me. I hope EMOTION the best for him.
Chandler PERSON : Kids, new dream...I'm in Las Vegasa GPE and I just want a million dollars MONEY !

I implemented Snorkel to label relationships between entities like people or organization. I took the following steps to apply Snorkel on my data.

For example, the text below is:

"Ross: I hate how you always do this, Rachel."

1. First step is to identify characters, which are 'Ross' and 'Rachel' here
2. Second is to define some relationship words between them, which is 'hate'
3. Third is to label relationship in the text, which is 'Negative'

So, by implementing Snorkel on my data, I was able to automatically label the types of relationships (positive or negative) between characters in the "Friends" series, making the analysis of interactions more structured and efficient.

Despite its overall success in identifying relevant companies, the model sometimes struggled. For example, it missed explicitly or implicitly stated entities.

For instance, if a dialogue says,

"We need to look into the new tech firm mentioned in the recent article"

- Model might fail to identify "the new tech firm" as a company because it's not named explicitly.

Furthermore, some objects may be misclassified because of overlapping references or similar nouns. These difficulties mean that although pre-trained models provide a strong starting point, additional steps such as adding domain-specific training data and resolving entity classes can lead to omitted information become accurate and precise.

For the extraction analysis of simply listed entities from subtitle data, one application that can extend these techniques to provide finer insights is in the area of supported summaries. By identifying key elements and their connections I can create short summaries of entire episodes or seasons highlighting key characters, places and events. This can be particularly useful for viewers who want to retell quickly or other audience members looking to catch important story points.

For example, imagine we are analyzing subtitles & it has entities like:

"Rachel," "Central Perk," and "wedding."

Using extraction techniques, I can identify these entities and then use them to create concise summaries. For instance, after processing the subtitle data above, I might generate summary like:

"In this episode, Rachel discusses her upcoming wedding at Central Perk, where the group reflects on their past relationships."

This summary highlights key characters (Rachel), places (Central Perk), and events (the wedding), making it easier for viewers to quickly grasp the main story points of the episode.

A powerful new application enhances the search recommendation system. By tagging subtitles and entities and their relationships, I can improve search algorithms to provide more accurate and contextual results. For example, **a search query for "Ross and Rachel's argument" can return specific scenes or episodes where these characters have notable interactions**, thanks to extensive entity tagging and recommendation systems that can use entity data to identify topics , color, layout or similar content, increasing user engagement and personalization.

Due to the complexity of human language, such as context, ambiguity, and different expressions, information extraction (IE) in natural language processing (NLP) is a practical task but a number of techniques can be used to simplify IE tasks and improve accuracy .

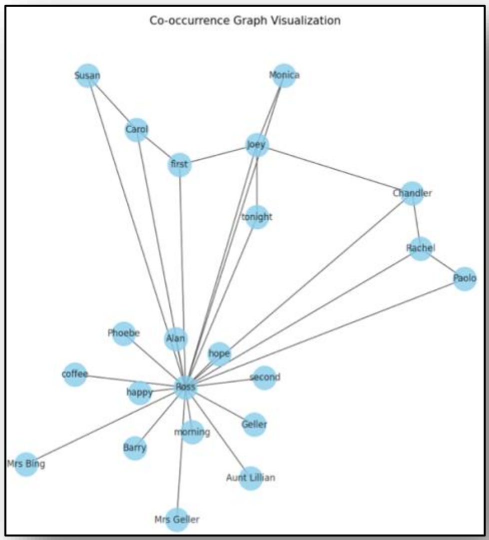
- One effective approach is to use **domain-specific** training data. Generic models do not work well on specialized texts because they lack the contextual knowledge necessary for accurate extraction. By creating and using data types specific to a domain of interest, such as legal documents, medical records, or TV show scripts, we can train auditory models of words, contexts about, and under the details of that particular domain

- Using **contextual embedding**, which capture the meaning of a term based on its context, can greatly improve the accuracy of named entity recognition (NER). Model like BERT provide embedding that considers the surrounding words resulting in good extraction and NER.

- Some annotation tools could make labelling data process more efficient. **Tools like Prodigy, Snorkel** facilitates the annotation process which allows domain experts to label data quickly and accurately.

I have also created a co-occurrence graph which visualizes relationships between entities based on how frequently they appear together in the text data.

1. Node represents an entity (such as a person, emotion, location, or other categorized item).
2. Edges (lines between nodes) indicate how often these entities co-occur within the same context or sentence.



In my dataset while analyzing dialogues from a TV show like "Friends," nodes representing main characters might be densely interconnected due to frequent mentions of their interactions. Entities like emotions or places may also show distinct clusters or connections, revealing thematic links in the text. I could see majority occurrence of 'PERSON' named entity (which makes sense because in every row(subtitle) of my dataset, there is a person's name).Overall, the co-occurrence graph is providing a visual summary that highlights prominent relationships and patterns among entities (mostly 'PERSON') in my text data, offering insights of how different elements interact and co-occur throughout the analyzed content.

Text Summarization

I used **LSA**, **TextRank** and **LDA** for text summarization. TextRank produced a high recall summary (0.71) but very low accuracy (0.15), showing that it captures many relevant sentences but also includes less relevant sentences. LSA achieved a balance of recall moderation (0.47) and precision (0.20), meaning that it provides a relatively accurate summary but still includes slightly irrelevant features. Whereas, LDA performed well with a precision of 0.30 and a recall of 0.34. providing a more focused summary but still with room for improvement.

The summaries generated using LSA and LDA methods align well with the key themes identified in the text analysis, as reflected in the slightly higher ROUGE-1 precision scores. These methods successfully gave the essential information, providing a coherent and relevant summary that matches the detailed content on the webpage.

ROUGE-1 Precision Scores:

- ***TextRank Summary:*** Precision 0.15
- ***LSA Summary:*** Precision 0.20
- ***LDA Summary:*** Precision 0.30

LDA is a useful method to discover hidden topics in large collections of text, making it easier to understand and organize the content. The precision score of LDA summary is 0.30. This means that 30% of the words in the LDA summary are also found in the human summary. This is the highest precision score among the three methods, suggesting that LDA was the most effective in identifying the key words from the text. The LDA summary performed the best at matching the human summary, followed by the LSA summary, and then the TextRank summary.

Top three topics from my topic model are –

- Topic 0: Keywords: how, david, sure, us, kiss, hands, aunt, my, hear, birds
    - This topic seems to include conversational and situational words.
- Topic 1: Keywords: um, way, max, just, nothing, thought, table, another, three, pick
    - This topic includes filler words and everyday conversational language.
- Topic 2: Keywords: i, ross, chandler, phoebe, rachel, monica, joey, oh, know, you
    - This topic features the main characters of the Friends series: Ross, Chandler, Phoebe, Rachel, Monica, and Joey.

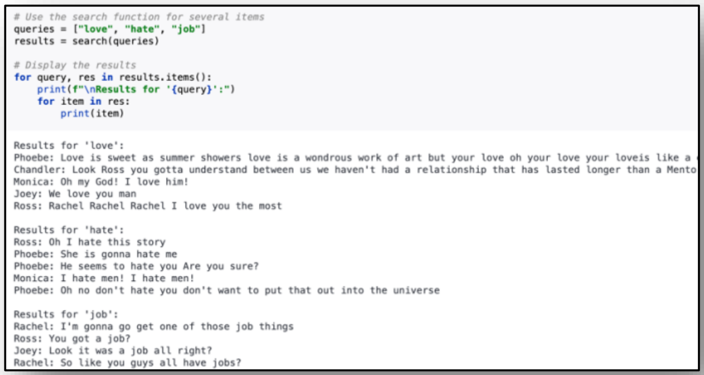
- Topic 0 captures specific events and personal interactions, providing insight into memorable moments and character interactions.
- Topic 1 represents the more mundane, everyday conversations, highlighting the show's focus on daily life and interactions.
- Topic 2 directly focuses on the main characters, which is central to the Friends series, summarizing their general interactions and relationships.

Although bag of words methods are useful for text summarization but they have few limitations. They are great in identifying key terms and patterns in the text by focusing on representing text data by breaking it down into individual words or tokens but they often lack the ability to capture the full context and nuances of the text, which in turn produces less coherent summaries and in some cases, the summary loses its meaning. Bag-of-words techniques like LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation) are useful for breaking down text into manageable parts and identifying key topics or terms. However, these methods can sometimes miss the finer details of meaning and how sentences relate to each other, which can affect the overall quality of the summaries they produce.

TextRank, another method, works well in some situations by ranking sentences based on their importance, but it can sometimes include less relevant sentences even though it captures a broad range of content. While these techniques are good for starting the summarization process, they often fall short in creating summaries that are both cohesive and contextually rich.

If the text is well-organized and the key points are structured properly in places, then grabbing sentences can be a good way to create summaries. This method can work well for simpler or shorter texts where the important information is straightforward to pick out. However, if the text is too long and complex then text generation technique might be a better option since these advanced model understands the context of the text and relationships between sentences, which helps in producing summaries that are not only brief but also smooth and coherent. For my subtitles dataset, I believe that text generation is a good option to get summaries because subtitles often contain fragmented sentences and informal language and text generation models are better at understanding the context and can produce summaries that are more coherent and meaningful. Also, text generation models are better in handling the informal and conversational nature of subtitles, whereas simple sentence extraction might not give much accurate results.

Enterprise search engines are useful for my work since they can index all the subtitles, making it easier to search for specific quotes or scenes quickly. I created a. Below is a snippet of search engine I created that has a search function which identifies and retrieves all instances or subtitles from the series where words like 'love,' 'hate,' and 'job'.



```
# Use the search function for several items
queries = ["love", "hate", "job"]
results = search(queries)

# Display the results
for query, res in results.items():
    print(f"\nResults for '{query}':")
    for item in res:
        print(item)
```

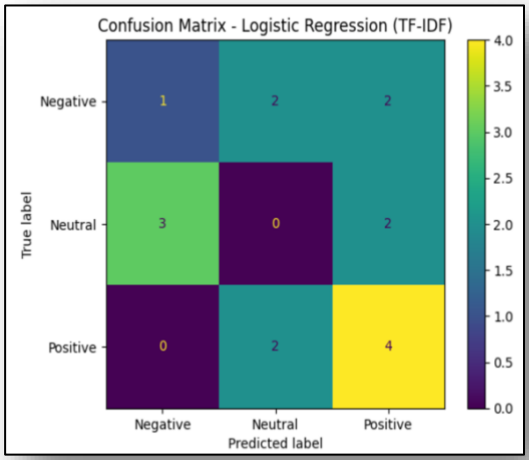
```
Results for 'love':
Phoebe: Love is sweet as summer showers love is a wondrous work of art but your love oh your love your loveis like a
Chandler: Look Ross you gotta understand between us we haven't had a relationship that has lasted longer than a Mento
Monica: Oh my God! I love him!
Joey: We love you man
Ross: Rachel Rachel Rachel I love you the most

Results for 'hate':
Ross: Oh I hate this story
Phoebe: She is gonna hate me
Phoebe: He seems to hate you Are you sure?
Monica: I hate men! I hate men!
Phoebe: Oh no don't hate you don't want to put that out into the universe

Results for 'job':
Rachel: I'm gonna go get one of those job things
Ross: You got a job?
Joey: Look it was a job all right?
Rachel: So like you guys all have jobs?
```

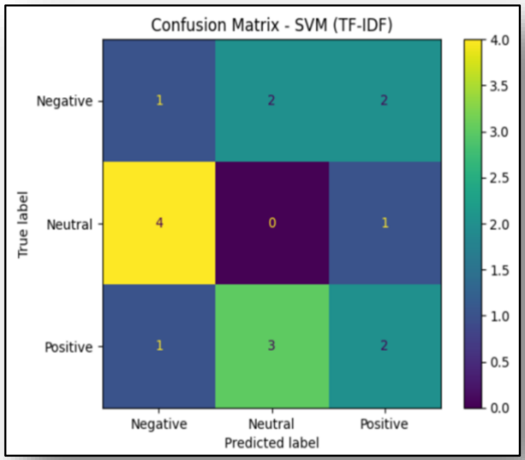
The result shows all the occurrences of ‘love’, ’hate’ and ‘job’ in the subtitles. Besides making searches more efficient, text summarization can be really useful in several ways. For example, I could create short summaries of each episode to give fans a quick overview of what happened. These summaries could also help me pinpoint key scenes or dialogues, which is great for putting together highlight reels or doing a more in-depth analysis of the show. Overall, this approach would make the content more accessible, inspire new content ideas, and even improve recommendation systems based on the show's events.

Classification

I performed sentiment classification using Logistic Regression and Support Vector Machine (SVM) models with TF-IDF (Term Frequency-Inverse Document Frequency) to extract features. I picked TF-IDF to extract features because it does a good job of showing text data. It points out key words in a document compared to a group of documents, which helps a lot in classifying text. TF-IDF is a preferred feature extraction when labelling a text based on words.

I chose Logistic Regression and SVM as my classification algorithms. Looking at the confusion matrices and classification reports, it could be observed that both models had a hard time classifying the "Neutral" class. In fact, neither model predicts any neutral instances. The Logistic Regression model has a bit higher overall accuracy (31%) than the SVM model (19%). The SVM model shows a better precision for the "Positive" class but doesn't do as well overall. Both models have low precision, recall, and F1-scores. This suggests they're struggling to classify the sentiment labels. The reason for this is because the classes in the data overlap or aren't distinct from each other. Moreover, as compared to SVM Model, the logistic regression Model has done a better job in categorizing the positive class, thereby it's f1-score being 0.57.





There are other things that I could have considered classifying my text:

- I might consider emotion detection, i.e. in addition to ‘positive’, ’negative’ and ‘neutral’ sentiments, I could consider several emotions like happy, angry, sorrow, frustrated or surprised. This could provide more granular insights into the emotional tone of dialogues.
- I might classify text based on the character speaking which could help in analyzing character-specific dialogues and their evolution throughout the series.
- Since "Friends" is a sitcom, classifying sentences based on whether they contain humor or jokes could help in studying comedic elements and their impact on the audience.

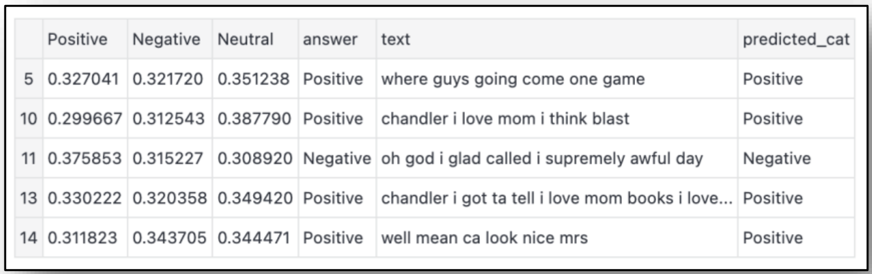
Since my data has subtitles from a series, **I think that the best option for feature extraction is TF-IDF** because this feature extraction works well in capturing the importance of words in the context of data thus balancing word frequency with document relevance. Moreover, TF-IDF helps in reducing the weight of common words across documents and emphasizing terms that are unique to specific classes, aiding in distinguishing between them. Since subtitles often include informal language and slang, TF-IDF can still identify significant terms and phrases by focusing on their relative importance rather than absolute frequency.

Word Embedding is also an importance feature extraction for text classification and could be used in analyzing datasets of subtitles (conversation/dialogues) since it captures deeper semantic relationship between words. It can capture synonyms and related concepts, making them suitable for tasks where understanding the meaning behind words is crucial (In my case I have implemented TF-IDF).

Explaining machine learning models is really useful since helps us understand why a model makes specific predictions, which is important for checking its accuracy and fairness. These explanations can reveal which features are affecting the results, helping us improve the model.

I have used **LIME** for this purpose. LIME explains the predictions of any ML Model by slightly changing the input text (like removing or swapping words) & then seeing how these changes affect the results. It interpret the predictions by highlighting which features (in this case, words or tokens) are most influential for a given prediction. It then builds a basic linear model around the specific prediction to help explain why the model made that decision.

Below table shows the result of LIME:



	Positive	Negative	Neutral	answer	text	predicted_cat
5	0.327041	0.321720	0.351238	Positive	where guys going come one game	Positive
10	0.299667	0.312543	0.387790	Positive	chandler i love mom i think blast	Positive
11	0.375853	0.315227	0.308920	Negative	oh god i glad called i supremely awful day	Negative
13	0.330222	0.320358	0.349420	Positive	chandler i got ta tell i love mom books i love...	Positive
14	0.311823	0.343705	0.344471	Positive	well mean ca look nice mrs	Positive

In the table above, for each sentence, the table displays the predicted probabilities for each sentiment category (Positive, Negative, Neutral). These scores reflect the model's confidence in each class. 'Answer' column has the true value of the sentence whereas 'predicted_cat' column has the predicted category of the sentence given by the model. 'Text' column is the actual text or sentence that was analyzed to determine its sentiment.

- For sentences like "where guys going come one game" and "chandler i love mom i think blast," the model is confident in predicting them as Positive with high probabilities for the Positive class.
- The sentence "oh god i glad called i supremely awful day" has the highest probability for the Negative class and is correctly classified as Negative.
- Sentences such as "chandler i got ta tell i love mom books i love..." and "well mean ca look nice mrs" are predicted as Positive, and these predictions align with the true labels.

We can implement LIME almost everywhere and see what the predictions are and to what extent they make sense. Imagine we have a model that decides whether a movie review is positive or negative. If the model says a review is positive, LIME can tell us that it's because the review used words like "great" and "amazing." On the other hand, if the review was predicted as negative, LIME might show that words like "boring" and "terrible" were influential in that decision.

These systems or techniques used to explain ML Model in text analysis like LIME, SHAP, Anchors etc. build trust with users by making the model's decisions more transparent. Additionally, if something goes wrong, explanation tools can help pinpoint whether the issue is with the data, the features, or the model itself. Overall, these systems are essential for ensuring models are clear, reliable, and effective.

To better understand our models and fix problems:

1. We can start by looking at the **errors they make**. This means checking the cases where the model's guesses are wrong to see if there are any common mistakes.
2. We can also create **visual charts** to understand how our data and model predictions look.
3. Trying out different features, adjusting how we **prepare the data**, or using different model types can also help improve results.
4. **Regularly testing and updating** the model with new information is important to keep it accurate.
5. **Talk to people** who know the subject matter well. Their insights can help identify why the model might be making mistakes.
6. We can identity which **features** (like words, phrases, or numbers) **are most influential** in the model's decisions. If certain features don't make sense, they might be causing errors.
7. **Test the model** multiple times with tricky cases or slightly altered inputs (more input data) to see how robust it is. This helps in understanding where the model might be vulnerable to mistakes.

Own Analysis

I aim to analyze the sentiment in the subtitles of the TV show "Friends" to understand character dynamics and interactions. To achieve this, I need to identify the most useful features of the dialogues. This can be done by comparing the frequencies of different types of features in the subtitles and evaluating different sentiment analysis approaches.

RQ1: Do the characters in "Friends" use positive sentiment words more frequently in their dialogues?

RQ2: Do antagonistic or conflict scenes in "Friends" feature a higher frequency of negative sentiment word?

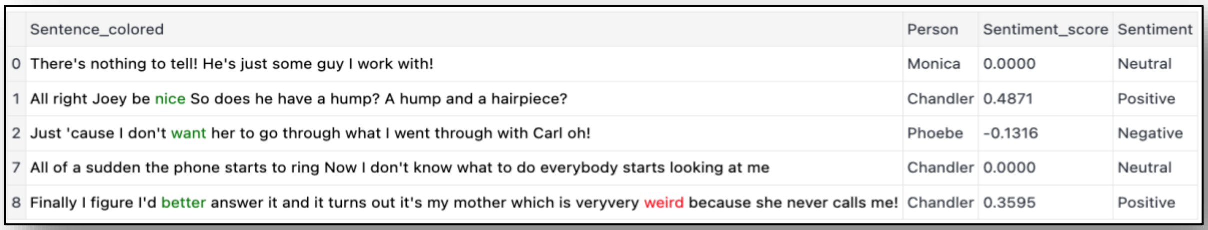
For the first question i.e. 'Do the characters in "Friends" use positive sentiment words more frequently in their dialogues?', I will analyze the frequency of positive sentiment words used by the characters throughout the show. The results will help determine whether positive language is more common in their interactions.

To analyze the sentiment in the subtitle, I focused on identifying the most useful conversational features to analyze emotions in subplots of the TV show "Friends" and understand character dynamics and interactions. I achieved by quantifying the frequency of various features a in smaller cases were compared and different sensitivity analysis methods were tested.

To address this research questions, I implemented two sentiment analysis methods:

- VADER Model
- SVM model with TF-IDF feature extraction and SMOTE for data balancing (I used SVM Model to enhance accuracy since VADER Model gave low accuracy)

I have used VADER (Valence Aware Dictionary and sEntiment Reasoner) Model to obtain sentiment scores for each dialogue in the subtitles. The sentiment scores of positive words was >= 0.05 and for negative scores it was <= -0.05, while for neutral words the score were 0. Positive sentiment words were labeled as 'green' and negative sentiment words as 'red'.



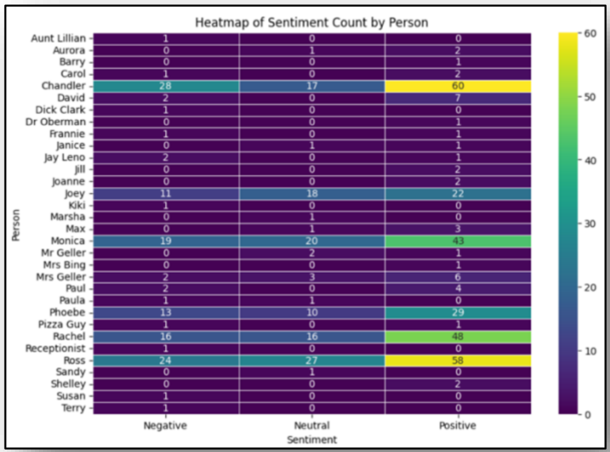
	Sentence_colored	Person	Sentiment_score	Sentiment
0	There's nothing to tell! He's just some guy I work with!	Monica	0.0000	Neutral
1	All right Joey be nice So does he have a hump? A hump and a hairpiece?	Chandler	0.4871	Positive
2	Just 'cause I don't want her to go through what I went through with Carl oh!	Phoebe	-0.1316	Negative
7	All of a sudden the phone starts to ring Now I don't know what to do everybody starts looking at me	Chandler	0.0000	Neutral
8	Finally I figure I'd better answer it and it turns out it's my mother which is veryvery weird because she never calls me!	Chandler	0.3595	Positive

In the table above, it can be observed that:

- Happy/positive/encouraging/supporting words in it then it is classified as 'Positive' sentiment.
- If a sentence has negative/sad words in it, then it is labelled as 'Negative' Sentiment.

- All the other sentences which neither have positive or negative words in it are labelled as ‘Neutral’ sentiment by the model.

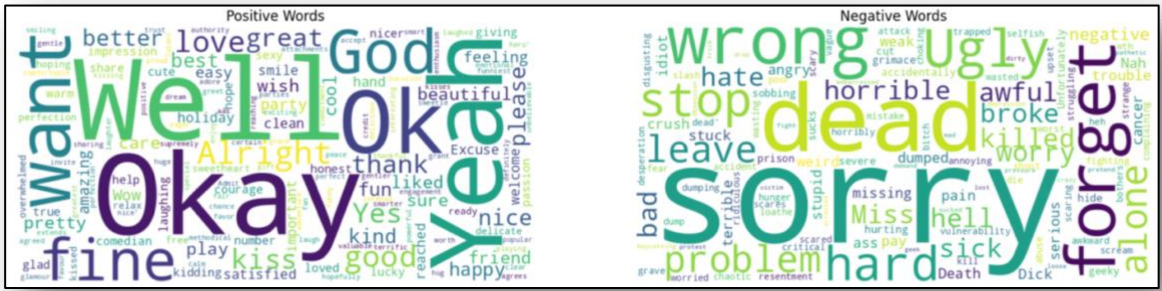
To further get more glimpse of ‘Positive’ and ‘Negative’ words, I employed heatmaps to visualize the sentiment distribution and the WordCloud library to illustrate the sentiments (positive and negative words).



From the heatmap, it is worth noticing that most characters, especially the main ones like Chandler, Monica, Joey, Phoebe, Ross, and Rachel, have a higher count of positive sentiment words in their dialogues compared to negative and neutral words. For instance:

- **Chandler**: 60 positive, 28 negative, 17 neutral
- **Monica**: 43 positive, 19 negative, 20 neutral
- **Joey**: 22 positive, 11 negative, 18 neutral
- **Phoebe**: 29 positive, 13 negative, 10 neutral
- **Ross**: 58 positive, 24 negative, 27 neutral
- **Rachel**: 48 positive, 16 negative, 16 neutral

I created a word cloud to better support my analysis which shows the occurrences/frequency of positive/negative words in the series indicating that this visualization highlights the diverse blend of emotions, revealing that the show is rich in both uplifting and challenging moments.



Thus, this answers my question stating that the characters in "Friends" generally use positive sentiment words more frequently in their dialogues.

To answer my second question i.e. ‘RQ2: Do antagonistic or conflict scenes in "Friends" feature a higher frequency of negative sentiment word?’, I examined scenes that involve conflict or antagonism and compare the frequency of negative sentiment words in these scenes to regular ones. The findings will reveal if negative language is more prevalent in moments of tension.

For this, I again used VADER sentiment analysis and word counting techniques. First, I applied VADER to each sentence in the dataset. This model assigns a sentiment score between -1 (most negative) and 1 (most positive). I then flagged sentences with a strong absolute sentiment score (greater than 0.5) as potential conflict scenes (since they contain most negative words).

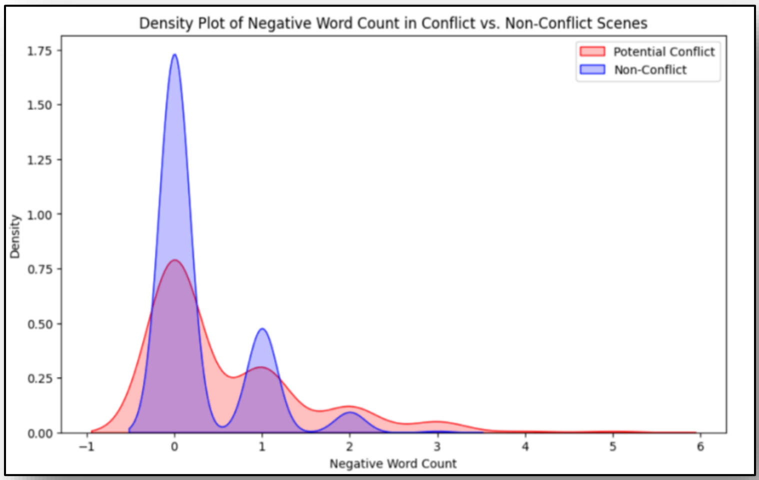
Next, I calculated the average sentiment score for both conflict and non-conflict scenes. The results showed that conflict scenes had a higher intensity, average score of 0.37 compared to non-conflict scenes, score of 0.06, reflecting that conflict scenes are more emotionally charged.

Afterward, I used an opinion lexicon, which contains lists of positive and negative words, to count the number of negative words in each sentence. By applying this to my dataset, I found that,

Average Negative Word Count:

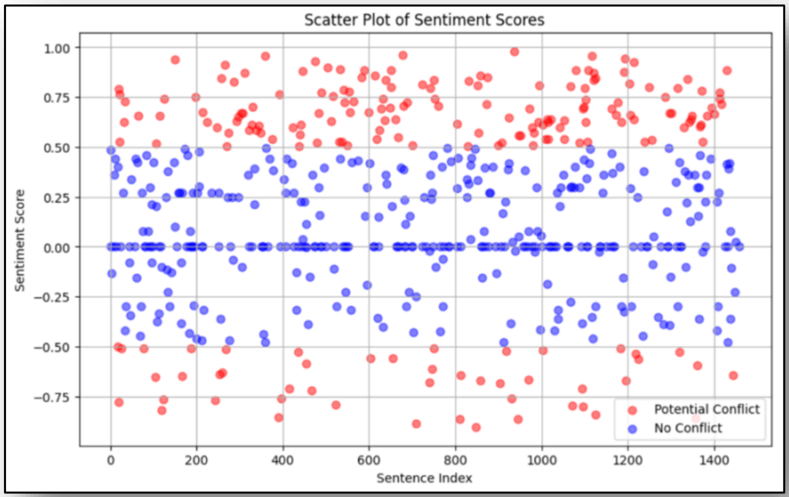
- **Potential Conflict Scenes:** 0.58
- **Non-Conflict Scenes:** 0.30

So, my analysis demonstrated that conflict scenes in Friends do indeed feature a higher frequency of negative sentiment words, aligning with the expectation that conflict would naturally involve more negative sentiment. This is reflected in both the sentiment scores and the negative word counts, which suggest that these scenes are linguistically more charged and reflect the emotional tension in those interactions.



Based on the above density plot and the average negative word count provided, I can interpret the results to answer the research question (RQ2) regarding whether antagonistic or conflict scenes in "Friends" feature a higher frequency of negative sentiment words. The density plot displays the distribution of negative word counts in both potential conflict scenes (red line) and non-conflict scenes (blue line). The plot shows that the negative word count tends to be slightly higher in potential conflict scenes compared to non-conflict scenes, though the distributions largely overlap.

Moreover, peak of the blue line (non-conflict) is slightly higher than the red line (conflict), highlighting that many non-conflict scenes have a lower or zero negative word count. Whereas, the red line (potential conflict scenes) has a more spread-out distribution, showing that there are more scenes with higher negative word counts in conflict scenarios.



The above scatter plot shows sentiment scores of each sentence (each dot is a sentence here). The red dot (sentence) has negative words in it and directs towards the likelihood of Potential Conflict whereas blue dot has positive words in it stating that there is No Conflict in those sentences (subtitles). Red points (potential conflict scenes) are spread across both positive and negative sentiment ranges, but there is a noticeable concentration in the positive sentiment area (above 0.5). Blue points (no conflict scenes) appear more frequently around the neutral sentiment score (close to 0) but also cover a range of both positive and negative sentiment areas.

The data suggests that antagonistic or conflict scenes in "Friends" do indeed feature a higher frequency of negative sentiment words compared to non-conflict scenes.

For instance, considering, a sentence like:

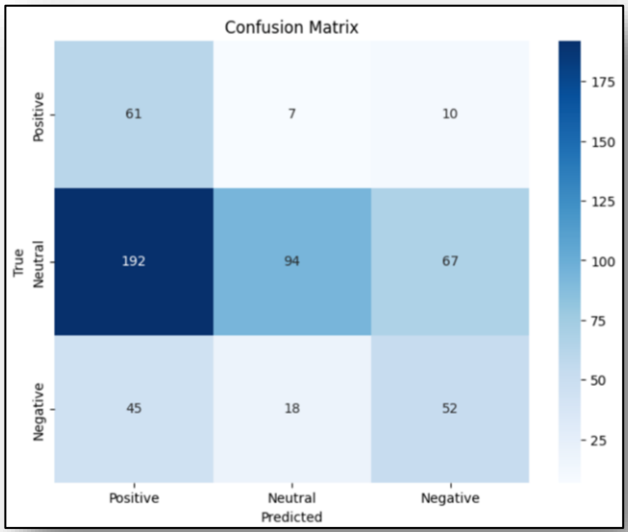
"***Just 'cause I don't want her to go through what I went through with Carl oh!***" with a negative sentiment score (-0.1316) may represent a moment of conflict, even though it's not extremely negative.

Meanwhile, non-conflict scenes tend to have lower or neutral sentiment scores, such as:

"Finally I figure I'd better answer it and it turns out it's my mother which is very very weird because she never calls me!" with a positive score (0.3595).

So, my analysis supports the hypothesis that conflict scenes are more likely to contain negative language, reflecting the tension or conflict in those interactions.

I then calculated the accuracy of the VADER Model compared to a Manually Labeled dataset (I labelled each subtitle as ‘Positive’, ‘Negative’ or ‘Neutral’ depending on my understanding) . I created a classification report and confusion matrix to assess the performance of the VADER model, using metrics such as precision, recall, and F1-score. The model has 38% accuracy in overall meaning hence the sentiment is right about 38% of the time. Positive sentiments have f1-score of 0.43 while neutral and negative sentiments have f1-scores of 0.40 and 0.32 respectively which shows performance disparities among the sentiment categories that imply some variations in the way they are expressed within this context.



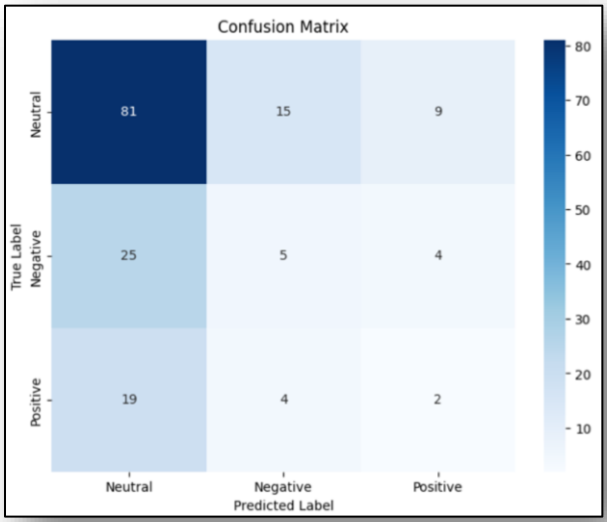
The low values of f1-score indicate that the model finds itself being caught between having a high recall or high precision particularly when it comes to null and negative sentiments expressions. This imbalance may suggest that even though it may be highly accurate on certain emotions, there are also many other misclassified emotions thus questioning its reliability.

In this confusion matrix, the highest value on the matrix (192) depicts incorrect predictions made by the model where Neutral sentiments were wrongly classified as Positive while on the other hand; the lowest indicates wrong predictions where Positive sentiments were classified as Neutral (7). Furthermore, number 94 represents those that were correctly labeled as Neutral which are correct but still need to be improved significantly because there is much room for development. From this information, it can be inferred that among all classes Neutral sentiments seem to pose a challenge to this model thus being misdirected as Positives most of the time or quite simply fail to detect them at all.

Since the accuracy of VADER Model is just 38% so, I implemented an SVM model with TF-IDF feature extraction (to see if I can achieve higher accuracy than what VADER Model gave) and applied SMOTE to handle data imbalance. I also evaluated the SVM model's accuracy against the same manually labeled dataset (which was compared against VADER), classifying sentiments into Positive, Negative, and Neutral categories.

These were the steps implemented which focused on Feature Extraction and Model Training:

- **TF-IDF Vectorization:** I transformed the textual data into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency) to capture the importance of words in the dialogues (labelling was based on words).
- **SVM Model:** I trained an SVM (Support Vector Machine) model on the TF-IDF features to classify sentiments into Positive, Negative, or Neutral.
- **Handling Data Imbalance:** SMOTE (Synthetic Minority Over-sampling Technique): I applied SMOTE to generate synthetic samples for underrepresented sentiment classes, ensuring a balanced dataset for training the SVM model.



I evaluated the SVM model's performance using accuracy, classification reports, and confusion matrices to understand how well it classified sentiments compared to the manually labeled data. I compared the performance of the SVM model with the VADER model to determine if the improvements in accuracy were significant. The classification report shows an overall accuracy of 54%, with the Neutral class having the highest F1-score of 0.70, indicating better performance compared to the Negative and Positive classes, which have much lower F1-scores of 0.17 and 0.10, respectively.

In the confusion matrix, the model correctly predicted Neutral sentiments most often (81 times), but struggled with Positive sentiments, correctly predicting only 2 out of 25 instances. This suggests that while the model does relatively well with Neutral sentiments, it has difficulty accurately identifying Negative and Positive ones.

Comparison of VADER Model against ML (SVM) Model with TF-IDF & SMOTE:

1. The SVM model shows better overall accuracy (54%) and excels in classifying Neutral sentiments as compared to VADER Model (38%).
2. The VADER model has better performance in classifying Positive and Negative sentiments.
3. The SVM model with TF-IDF and SMOTE offers a more balanced weighted average F1 score, indicating better performance across all classes.
4. SVM clearly performed better than VADER because of TF-IDF feature extraction & SMOTE
5. I also could have experimented with different feature extraction methods or combine multiple features (e.g., TF-IDF and word embeddings), further tuning of the SVM hyperparameters (C, gamma, kernel) which could have potentially improved both Model's performance.

I was interested in running this analysis because "Friends" is a classic show that has captured the hearts of viewers for years, and I wanted to dive deeper into what makes the characters and their interactions so memorable. By looking at how often the main characters use positive or negative words compared to secondary characters, I hoped to uncover patterns in their relationships and personalities. This analysis could also reveal how these patterns affect the audience’s feelings towards different characters, making us understand why we love or dislike certain characters so much.

In addition, I wanted to see how moods changed in different places, especially moments of conflict or hostility as opposed to regular everyday moments. "Friends" is known for its humor, but it also has tense and tense moments. By studying how moods change in these situations, I want to see how writers balance emotional intensity with light-heartedness, how this contributes to the overall appeal of the show for insight into what makes TV a program is compelling and relatable, as well as using sensory development to create compelling stories.

Additional Analysis - BART Model

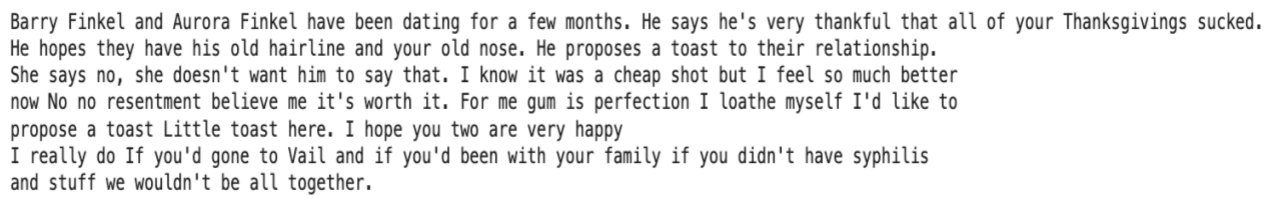
I aim to develop an abstractive summarization model for the subtitles of the TV show "Friends" to generate concise and coherent summaries of episodes for Analysis Purpose.

I have used the pre-trained **BART** model to effectively generate concise and coherent **abstractive summaries** of lengthy dialogue scenes from the Friends series, capturing the essential information while reducing the text length significantly.

Implementation of BART on my text :

1. **Filtering Long Sentences:** Selecting subtitles that are longer than 250 characters (so that the model runs smoothly).
2. **Combining Text:** Merging these selected subtitles into a single string.
3. **BART Model Flow:**
   - Import libraries from transformers package -
     - BartTokenizer - encoding and decoding text
     - BartForConditionalGeneration - BART model used for generating summaries.
   - Load Pre-trained BART Model and Tokenizer
   - Tokenize and Encode the Text
   - Generate Summary
   - Decode the Summary

After implementing BART on my Friends series dataset (subtitles), I got the below abstractive summary:



```
Barry Finkel and Aurora Finkel have been dating for a few months. He says he's very thankful that all of your Thanksgivings sucked.
He hopes they have his old hairline and your old nose. He proposes a toast to their relationship.
She says no, she doesn't want him to say that. I know it was a cheap shot but I feel so much better
now No no resentment believe me it's worth it. For me gum is perfection I loathe myself I'd like to
propose a toast Little toast here. I hope you two are very happy
I really do If you'd gone to Vail and if you'd been with your family if you didn't have syphilis
and stuff we wouldn't be all together.
```

Wrap Up

The use of NLP (natural language processing) techniques for text analysis, as seen in this review of "Friends," highlights their wide applicability and effectiveness in a variety of textual contexts Whether it is TV whether assignment writing, social media posts, or literary industry reviews, NLP can reveal hidden patterns, emotions and relationships that may not be immediately obvious. Even basic techniques such as sentiment analysis and frequency counting can yield valuable insights into how language is used, emotions are expressed, and narratives are structured While more complex models can provide more nuanced explanations for him, this study shows that even simple NLP techniques are powerful tools for manipulating and understanding information in different contexts, making them important in both academic research and real-world applications.

Apart from implementing NLP techniques on subtitles data, these techniques could be implemented in a wide range of domain. For example:

NER could be implemented in:

- Finance to extract company names and stock symbols from news articles to trach market trends.
- Healthcare to extract important medical terms from clinical notes, electronic health records, and research papers.
- Customer Service to enhance customer support by extracting key information from customer queries and complaints, such as product names, issues, and customer details
- Education to help in analyzing educational content to identify key concepts & institutions which can indeed support curriculum development, academic research.

But NER can have trouble with words that have multiple meanings, different ways of naming things, and context-specific meanings. It can also be challenging to get accurate results when dealing with different languages and formats. It can also struggle with different languages or informal writing styles, like slang or typos.

Information Extraction could be implemented in:

- Legal document analysis to extract important details from legal documents, such as case names, court decisions, legal statutes.
- Travel to fetch travel reviews, booking details, and social media mentions.
- E-commerce to extract product information, customer reviews, and feedback from various sources.

Information extraction can be tricky because the way information is presented can vary a lot. It often needs advanced tools and algorithms to accurately understand and pull out the right details from the text.

Text Summarization could be implemented in:

- Healthcare as it can condense medical records, research papers, and patient histories into concise summaries, making it easier for healthcare professionals to access and review important information quickly.
- Scientific Research for summarizing scientific papers, reports, and articles can make it easier to review and comprehend large amounts of literature.
- News Media for summarizing news articles and reports to helps readers get the main points quickly, keeping them informed without having to read full-length stories.

Some challenges of text summarization could be to pick out the most important information without losing important details or changing the meaning (in my Friends dataset, although text summarization like topic modelling was to summarize the larger body of text while retaining its key information and meaning however some of the text summarization techniques failed to capture the emotions & nuances of the context because of which the summary didn't hold a true accurate meaning).

Text Classification could be implemented in:

- Legal to categorize legal documents, case law, and contracts into relevant categories, such as types of cases or legal issues. This helps in organizing and retrieving documents more efficiently
- Finance for tasks such as fraud detection, sentiment analysis of financial news, and categorization of transaction types.
- Education to classify (label) academic papers, essays, and other educational content by subject or grade level which could enhance the process of manual categorization.

Classifying text accurately can be tough because language varies a lot and the meaning can be subtle. It needs clear categories and good training data to work well. Sometimes, categories can overlap, making it hard to decide which one a piece of text fits into.

As such, complex models can also be of help in transformation and comprehension of text, most especially with large data or nuanced language, for example, sarcasm, context, ambiguity. Of course, this could include very simple models too, but it is just what it is: simple, taking into account that the task may regard plain text or may only seek to obtain a general view of it fast. The choice between complex and simple models will depend on the task at hand, quality of data, and how much accuracy or detail one wants. Even though complex models, such as deep learning, can expose these more intricate patterns, they will be expensive in computation costs and may even lead to non-explainable models. So, often it is beneficial to start with simple models and then move towards more complex ones if required.