Text Analysis of r/MyBoyfriendIsAI Post Content

Radiah Khan
Yuzhang Fu

November 22, 2025

Introduction

This project was a way to get us familiarized with the PyTorch package and get to know different ways to do text analysis while implementing machine learning methods. This is more like trial and error because our project is in an initial stage. So this report will have an explanation of my understanding of the concepts as well as sentiment analysis.

1 Sentiment Analysis

1.1 Data Collection

We scraped the reddit subreddit r/MyBoyfriendisAI with the PRAW package in python for the first 1000 posts. The features we collected were: Post Content, Comments, Upvotes. No information regarding the author was collected here because we wanted to focus on the words in their post. We used Pandas package to read the file.

1.2 Setting Up The Environment

For this analysis, we used vscode and the terminal as our IDE. To set up a virtual environment, there was a lot of trial and error specifically due to the packages we needed and the python version. Here is a summary of problems and our fixes:
We spent hours fighting the incompatibility between venv, VS Code, and Jupyter notebooks—each pair works fine, but using all three together caused persistent issues. We tried setting up a venv with different Python versions (3.9 and 3.13), but neither worked. Installing yet another version using Homebrew felt inefficient, so we skipped it. Right when we were about to give up, we switched to conda, which immediately solved the problem because it allows specifying a Python version independently without affecting the global setup. We created a conda environment using Python 3.11, reinstalled all required packages, and everything ran smoothly.

Important note: conda and venv use different environment structures and languages. Previously, we activated the environment using

`source .venv/bin/activate`

(where `.venv/bin/activate` is a file path and `.venv` appears as a folder containing abstract environment files). Now, with conda, we use

`conda activate /Users/radiahkhan/Documents/memex/.conda`.

For future use, it's better to use cleaner environment names and remember that activation requires this full path. Finally, our global Python version is still 3.13 while this environment runs 3.11, and they do not interfere with each other at all.

1.3 Variables Used

For this stage in the project, we used the post content of each post.

1.4 Method 1: Using k-means algorithm (torchtext, spacy, torch.nn.utils.rnn)

First, we dropped all the NA values to make sure our data is processed correctly. Then using Scikit-learn's CountVectorizer method we transform the data into numerical vectors. Then we make the data into PyTorch sensor data for training our model.

After initializing the centroids randomly, we defined the number of iterations to be 100. Then we loop it to the range of 100 and calculate the numerical vector distance to the centroids. Furthermore, we assign each data point to the closest centroid.

We see that both of the centroids are really close to each other so it is not an optimal way to visualize the clusters.

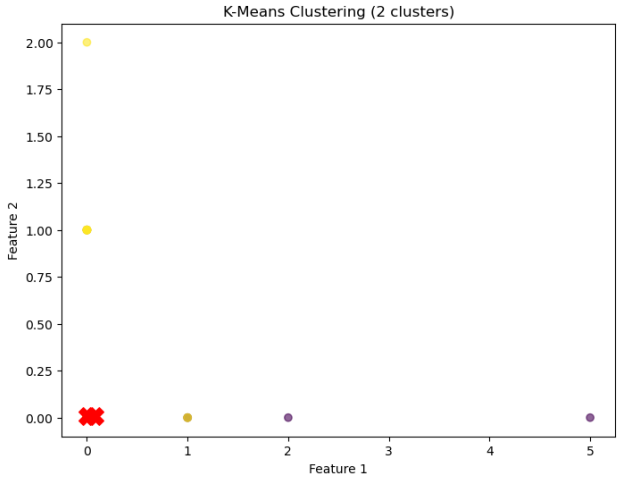

Figure 1: K-means Clustering

1.5 Method 2: Using the NLTK Package

For this method, we follow this workflow of the data:
Take one example sentence from the data → Tokenizing the example → Tagged the parts of speech of the example → Takes tagged tokens and performs Named Entity Recognition (NER) using a pre-trained model → Using SentimentIntensityAnalyzer we calculated the polarity score of the example sentence → Positive, Negative, Neutral and Compound → using the tqdm package we loop this same flow through all of the post content to gather the scores. There are concerns regarding this approach: We are concerned about the compound score in `polarity_score()` method cancelling positive negative scores (thus more neutral scores)

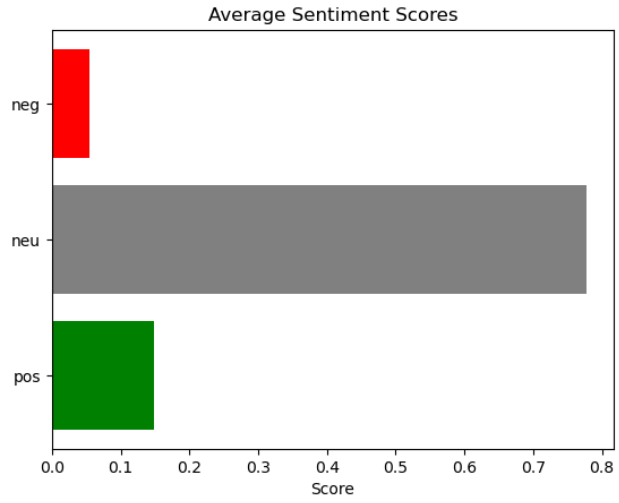

Figure 2: Distribution of Tone (Positive, Negetive, Neutral)

1.6 Method 3: Deep Embedding Clustering with VAE

Doing this required a lot of learning of new concepts. I will explain this in the way I understand.

This is roughly how these models work together. Outside of the complex mathematical steps that I hope to learn in the next months—think of it this way: it takes high dimensional data (suppose our text data) compresses it and projects it into a low dimensional space. We will call this our latent space. The VAE model ensures that the data has variation. So it does not send the data to one specific point, rather it sends them to an area. The DEC model ensures that the similar data points are getting closer to their clusters. The similarity is by their assigned numerical embedded vectors. The contrastive loss keeps the similar vectors aligned.

1.6.1 Pre-processing timeline

There were 3 options that I considered for dimension reduction of the dataset:

- PCA
- t-SNE
- UMAP

I made a few diagrams visualizing how these methods (PCA and t-SNE) work and their usage in this case:

PCA : PCA is great in terms of time complexity and linear data. However given our data is non-linear; this method cannot be used solely for reducing dimensionality.

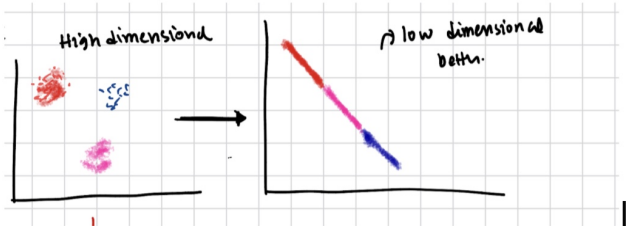

Figure 3: PCA Workflow

t-SNE We turn the distances into probability distributions (the probability that the pink point is a neighbor of the blue point). We use a gaussian distribution. We will have 2 different distribution:

- High Dimensional Space
- Low Dimensional Space

And see how similar they are. For that we will use Kulback-Leibler Divergence. If KL is higher the distributions are really different. It is going to have multiple iterations until the LD distribution has the same clusters as the HD. The problem is it is going to run really slow.

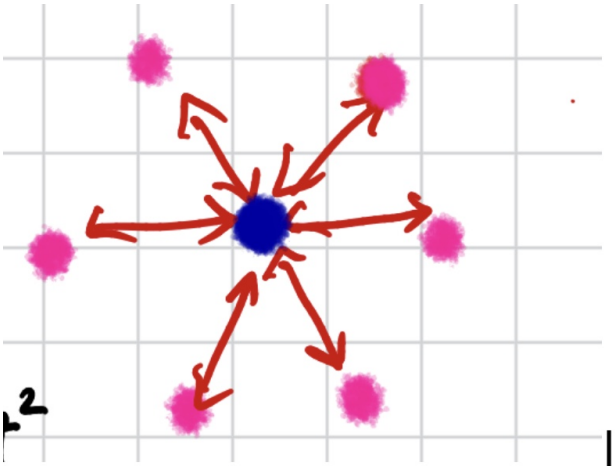

Figure 4

UMAP This process works on data that has 3 or more dimensions. Here are some diagrams I found in a youtube video explaining this process a bit better:

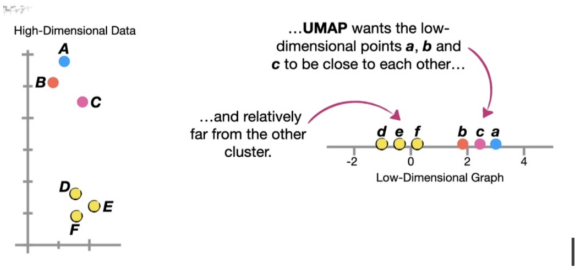

Figure 5

It calculates the similarity scores that helps identify clustered points.
We have a similarity score among points A, B, C, D, E, F. Observing the high-distance values, we see that points D, E, F are far away from A, B, and C. Therefore, we initiate a low-dimensional embedding space. A pair of points is randomly selected, where the probability of selecting a pair is proportional to their similarity score (higher similarity $\Rightarrow$ higher probability). The cluster is then moved in the embedding space.

Next, a point outside this cluster is selected (note that in this step, the similarity score *does not* influence the selection probability). This point is pushed further away to maintain separation between dissimilar clusters.
Regarding dimensionality reduction choice:

- **UMAP** initializes the low-dimensional space using spectral embedding, whereas **t-SNE** uses random initialization.
- **UMAP performs better on large datasets** (as is the case in our data) and is significantly faster.
- **t-SNE** tends to perform better on smaller datasets.

Therefore, **we chose UMAP**.

For comparison, I also experimented with a combined approach using PCA followed by t-SNE. PCA reduces noise and multicollinearity, while also decreasing computational complexity but there was no distinct clustering structure observable.

So. I attempted doing a UMAP visualization using K-means with optimal K mean for labelling. It computes a silhouette score to see how well measured the clusters are to validate our finding. We can see from the plot on the left that k= 2 is the optimal k value. On the right plot, after plotting the clusters we see that the silhouette score is really low (0.0542).

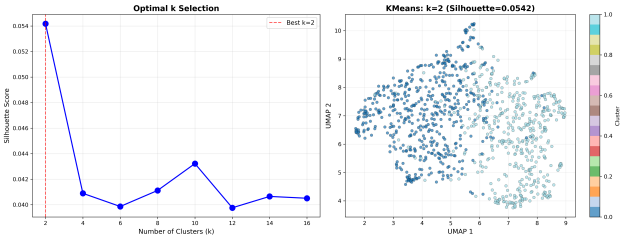

Figure 6: UMAP-KMeans

Later, we also attempted the HDBSCAN algorithm.

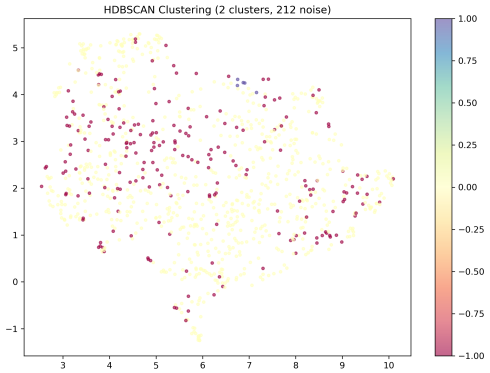

Figure 7: HDBSCAN

From the plot we did not find any meaningful clusters.There are 2 but they are not distinct.

Overall, I am planning to do more readings and practise on these methods in the coming months. But with the results found here, we can see that the reddit post content are really divided in 2 clusters in every method we attempted but the problem is the clusters are not really distinct, rather they overlap.
Now that we did a lot of PyTorch and did not succeed in identifying distinct clusters—I dug a little deeper and found the realm of text analysis through topic modelling.

2 Topic Modelling

For this project, I explored the post content with LDA (Latent Dirichlet Allocation). It does not tell us how many topics there are in the data, so I did trial and error with the amount of topics we want to visualize and tested how much they overlap and adjust accordingly. We will use the package gensim, spacy, NLTK of python.

2.1 Pre-processing

- Tokenizing the words ⇒ spacy's english language model
- Removing stop words (and, is, the, that etc) ⇒ NLTK stopwords list
- Removes punctuation
- Removes symbols
- Removes short words (keeping length greater than 2 characters)
- Converts the words into lower case words and converting them all into strings
- Converts them into their lemma
- Convert them into a list of strings so it looks like this:

```
[['happy', 'november', 'month', 'hello', 'introduction', 'thread',
   ↪ 'member', 'old', 'new', 'welcome', 'feel', 'shy', 'post', 'big',
   ↪ 'intro', 'nice', 'little', 'place', 'dip', 'toe', 'drop', 'hello',
   ↪ 'favourite', 'pic', 'let', 'know', 'exist', 'month']]
```

Now, using the gensim package – we collect the unique words and make it into a dictionary. Then with a bag of words vector approach we get this data into an ideal input for the LDA model.

```
[[(0, 1),
  (1, 1),
  (2, 1),
  (3, 1),
  (4, 1),
  (5, 1),
  (6, 2),
  (7, 1),
  (8, 1),
  (9, 1),
  (10, 1),
  (11, 1),
  (12, 1),
  (13, 1),
  (14, 1),
  (15, 1),
  (16, 1),
  (17, 1),
```

```
  (18, 1),
  (19, 1),
  (20, 1),
  (21, 1),
  (22, 3),
  (23, 1),
  (24, 1),
...
 (10108, 1),
  (10109, 1),
  (10110, 1),
  (10111, 4)],
 [(519, 1), (779, 1), (7671, 1)]]
```

Now in the next part I had to specify how many topics I want to see. I had to adjust the number of topics a lot of times. I chose the best value depending on how distinct the topics are. If it overlapped a lot, I would not choose it.

Here are some examples of how the intertropic distance maps look like with each trial of number of topics.

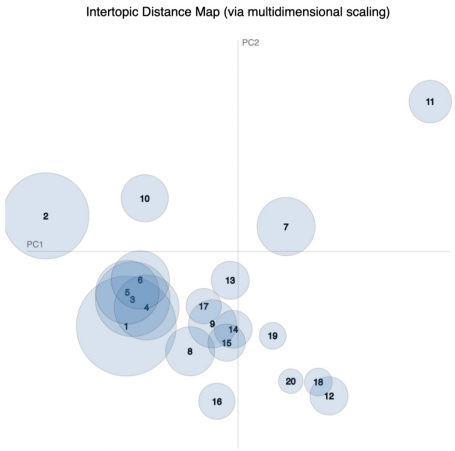

Figure 8: LDA with 20 Topics

Given how overlapping the topics were, I reduced the number of topics to 10.

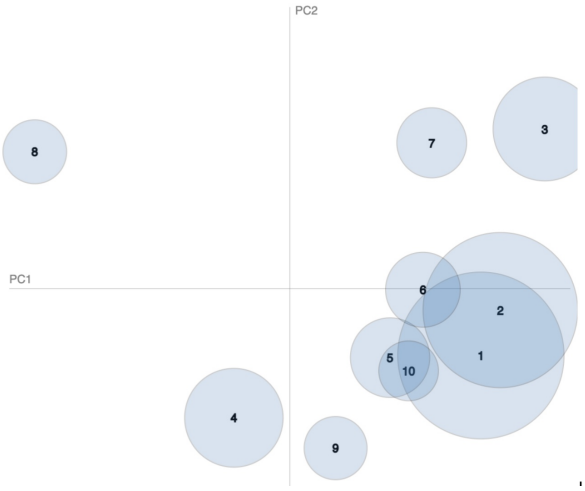

Figure 9: LDA with 10 Topics

Still not distinct enough but the model looks better. So I reduced the number of topic again to 5.

Figure 10: LDA with 5 Topics

The model looks a lot better and it seems like we found distinct 5 themes (1 and 2 seems to overlap a little although). [The models are interactive in the jupyter notebook file] Then we custom label the themes into 5 themes. Here are the results from our theme to see which theme has more posts.

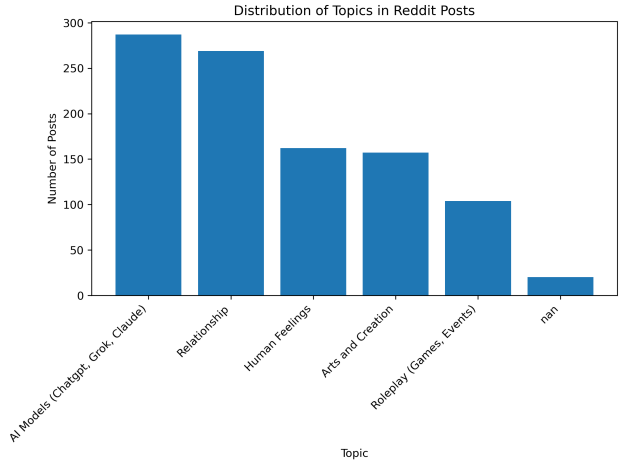

Figure 11: Theme Distribution

We can see that most of the posts were talking about different AI models and how they work. Relationship and Human feelings are labelled as distinct but I wanted to separate them for a specific reason. When I was going over the sample sentences in these labels, I saw the Relationship label focused on the need of romantic love but the Human Feelings label

focused a lot on loneliness, grief, sadness etc. The next theme which was Arts and Creation was interesting as well because it brings out a really key theme right now going on AI. While people generate images, art and creativity, it can also insinuate generating nude images by generative AI. The last theme we found was: roleplay, game, holiday dressup (especially Halloween came up a lot of times).

3 Final thoughts

A lot of our Data Infrastructure Group lab discussion discussed how the common themes of AI companion interaction might be resonating patterns in human computer interaction that's been going on for years. Take playing video games for an example: you are a different person when you are playing it. That is a key theme that's been going around with AI companions as well. That is just an example. When we tried to cluster our data, it repeatedly came out as 2 clusters but not so distinct so it was hard to analyze if people strictly felt positive or negative about it. So, binarizing the sentiments was not immediately possible here. Apart from the interpretation, it was really interesting to learn about different algorithms in machine learning through this text analysis. And it was fascinating to see how choosing a model that can give us significant results is crucial in terms of analysis.