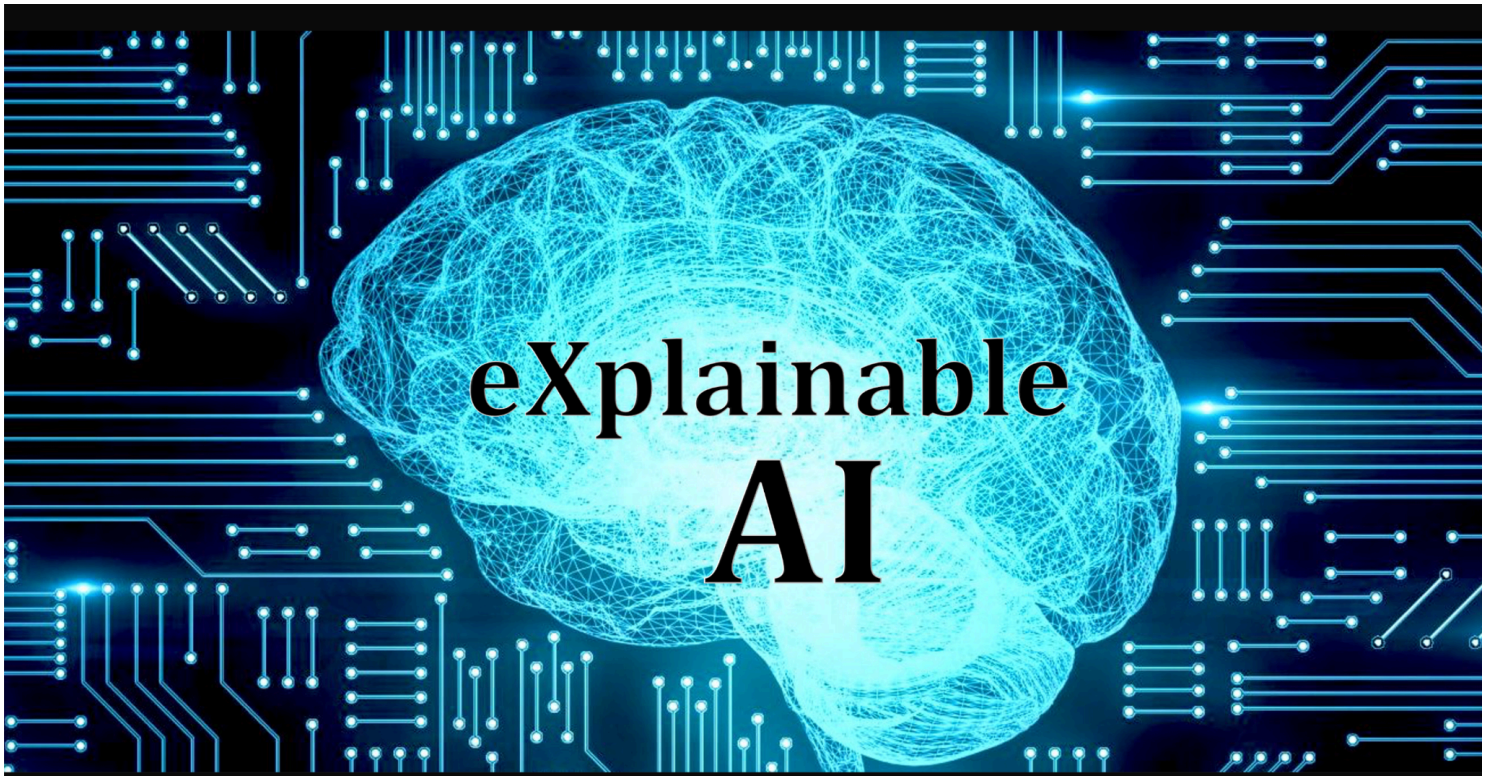


Deep Understanding Of Models

17.04.2024

Prakhar Sinha
Grid Dynamics
Silpa Gram Craft Village
HITEC City, Hyderabad

Objectives:

We will try to perform 3 main tasks:

1. Make an API and secure it.
2. Perform model Inferencing to see which features result in the solution
3. Perform grouped inference and see the seperativity of the different classes.

Previous Project Recap :

In our previous project, we made a query classifier that takes in questions and images as prompts and classifies them into subclasses, which would help in model specialization. We also saw that we had very high testing accuracy of around 92–93% for each class, and now we can try to explain which words have the most influence in the decision-making process and see grouped inference too. But before that, we attempt to make a secure AI. We will be using FastAPI.

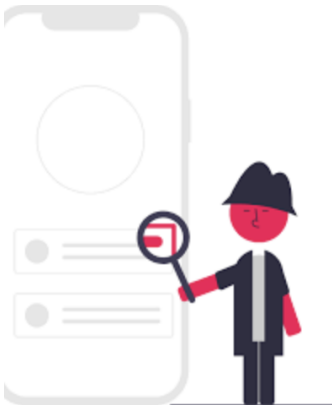
Secure API creation:

Question Prompt Check

In our first task of secure API creation, we use the FAST API. First, we will validate the inputs. We check if the question prompt given has words with the right spelling. Our sentence_checker.py file does that. It uses the NLTK dictionary, which has all the valid words. It checks if all the words in the user-given question are present in the dictionary. If not, it will not perform prediction and give an "`invalid sentence.`".

Image Link Check

For checking the image link, first of all, we check that the link is “https” and not “http”. If it is not https, we return `"insecure link."`.



Then we check that the image link given by user is in image format. Our model supports

jpg, jpeg, png, and CAU formats. So we check that the link belongs to one of these. If not, it will output `"invalid link format."`.

If both of these are valid, it will return `"Valid Input."`. After that, it will pass the prompts to the model. This way, we ensure invalid prompts are not passed to model

Making API key

A JWS token would be generated while our secure_api.py file was being executed. This question will appear in the API; if the user enters the token incorrectly, the program will not run.

We have set the token's validity to one hour. Seeing as how much time users spend on average, we can also cut it down to thirty minutes.

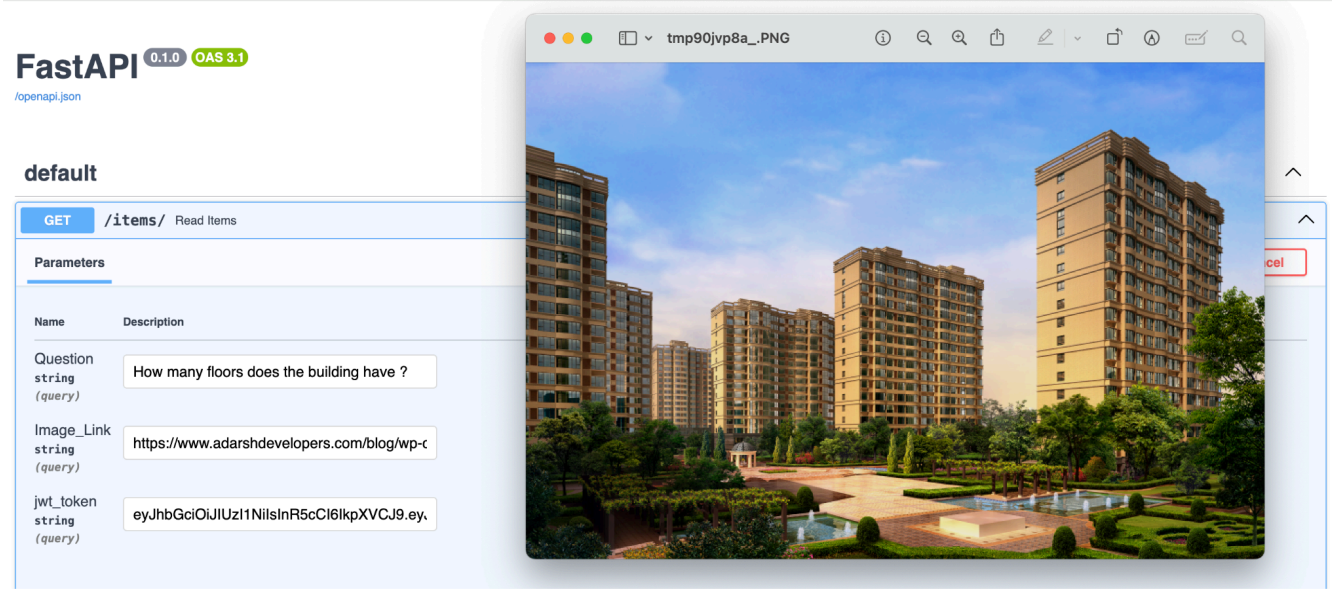
This is an illustration of a JWS token.

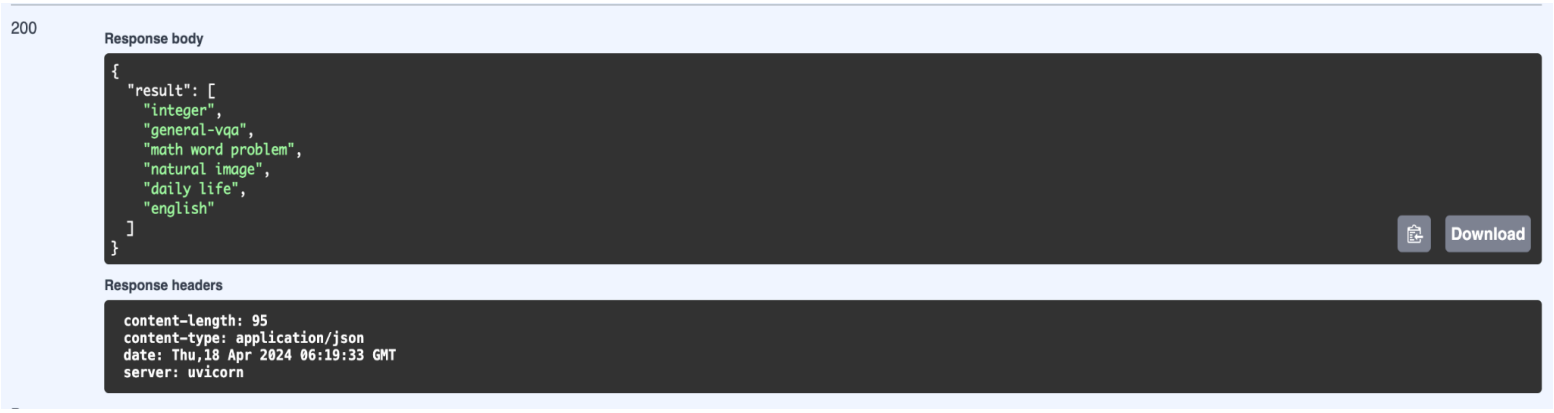
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxMjMsInVzZXJuYW1lIjoiZXhhbXBsZV91c2VyIiwiZXhwIjoxNzEzMzQ4OTA2fQ.8qTh13P2yOGdG6bsSKudnUpE8Mh22Q7n5FApKwwk4IQ

This offer is only good for one hour. To ensure that it is authenticated, the user must receive this from the file owner.

Test Example:

I have now tested my API. In the interface, I have put the question as “How many floors does the building have ? “ This can be a very tricky question, even for best models like GPT-4. If we are able to assist with the answer by classifying, it would be a great help.





We can see that it is correctly classifying the question as answer type integer, general visual question answer, math problem, natural image, daily life problems, and English. All the classifications except the math word problem are correct. Please see the Project-2 repository on the same GitHub for more information about the various classes that are available and the accuracy of my model.

Performing Model Inference:

Here, our goal is to determine which words in the prompt have more influence on decision-making. We establish a measure to examine the impact of language on judgment.

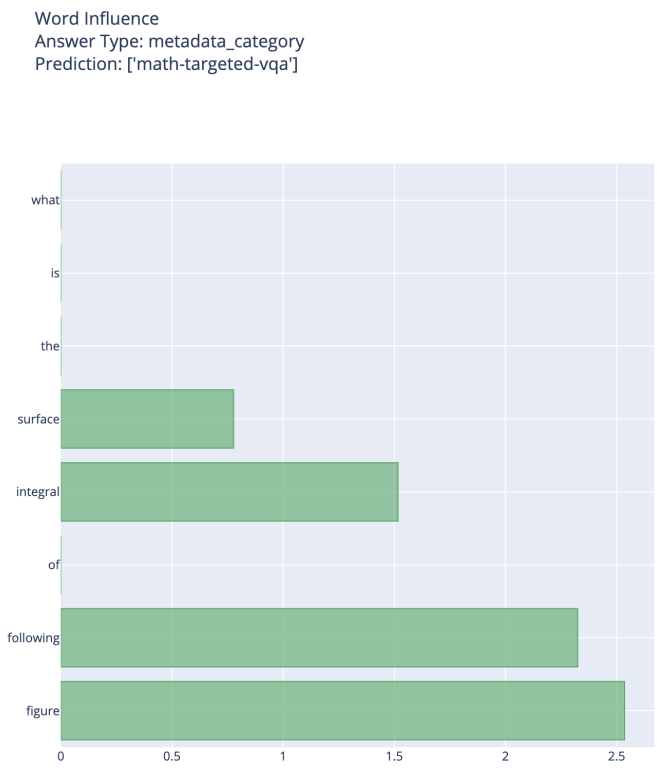
For that, we would use the coefficients from SVM and the TF-IDF vectors. The coefficients in SVM tell whether the word has a positive or negative influence on the decision-making process. A negative value of the coefficient would mean that the particular word is giving the prediction in the wrong direction. A positive value would mean that it is helping positively in classifying the current task.

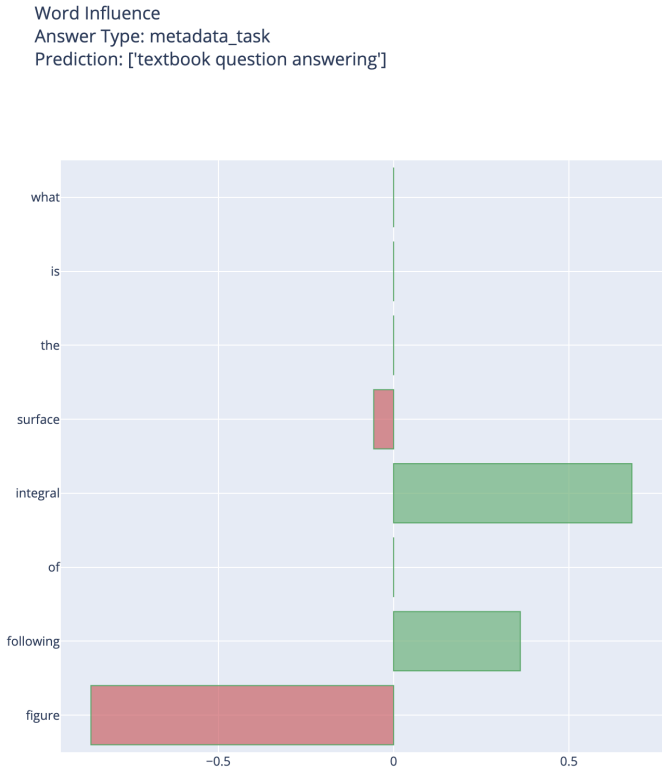
The TF-IDF vector shows how important the word is in the sentence. For example, words like “the” and “he” have very little meaning and provide less information. We combine both of the above parameters to check the word influence. We also scale both parameters before combining to make sure that one of the parameters does not have a bias. We get the influence of each word in the sentence in the following format:

[This is a cat sitting on a mat]

[0.001, 0 ,0 , 0.05, 0.1, 0, 0, 0.08]

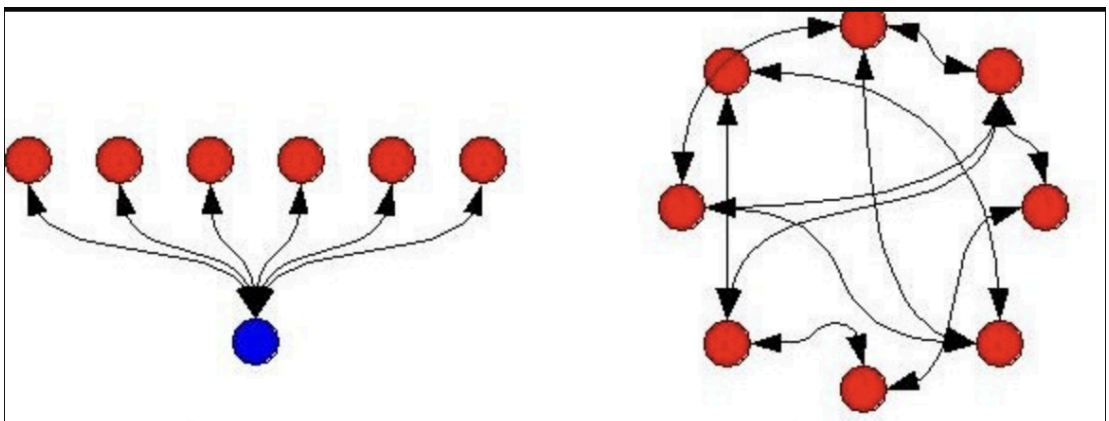
Now we can also plot sentences across different classes of previous projects to determine the influence. For example, we can try for the sentence “ What is the Surface Integral of the following figure ? “





I have shown the two classes' graphs. We can see that words like surface and integral have a good influence on determining the class as a math-targeted visual question answer.

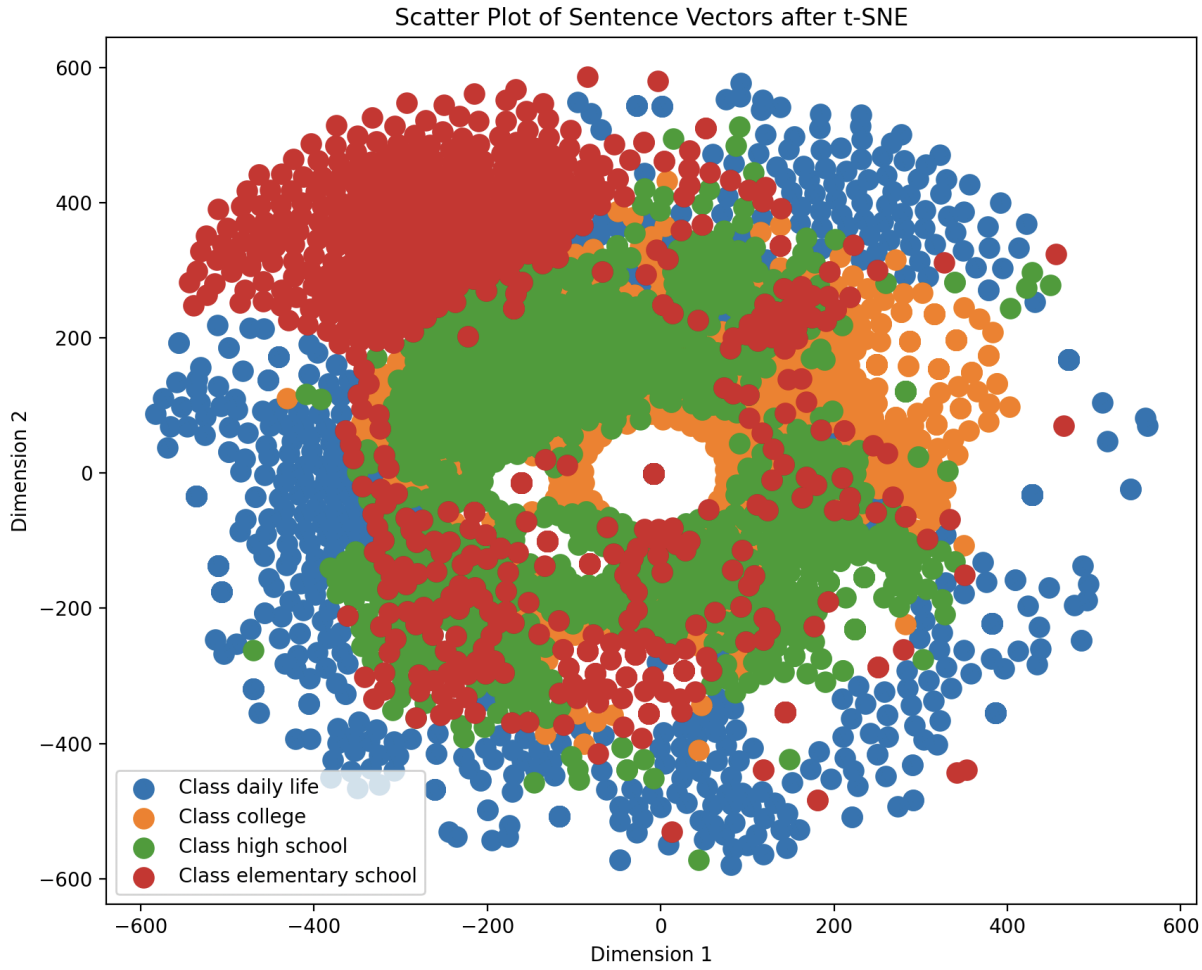
We can observe from the second graph that the word, say "figure" has a negative impact on whether or not it is classified as an answer to a textbook question. which is accurate given that the word "figure" will attempt to place it under the "Figure Question Answering" category. It is correctly classified as textbook question-answering, despite its high influence. This is a result of SVM's decision-making process, which considers the impact of each word on each other. This explains why, even when we use words that have a negative impact, our predictions are so accurate. The choice was made by taking into account every word collectively rather than each word separately. This explains why the results we get are so accurate and unaffected by a few unrelated words.



Performing Grouped Inference:

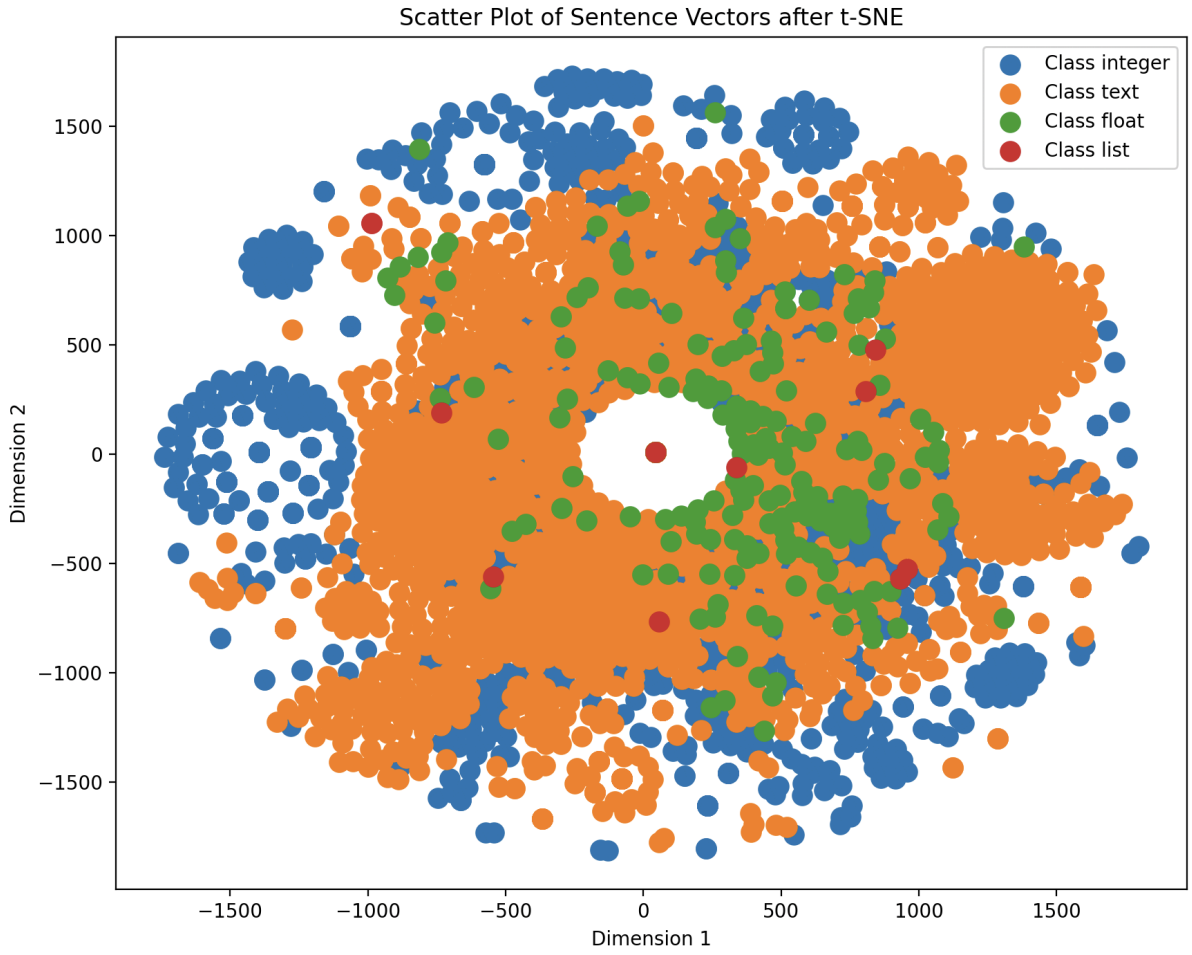
I will now show the grouped inference in the model. For us to see the grouped inference, we are going to try to perform dimensionality reduction to 2 dimensions and then plot it to see the seperability or clusters present. We have many different classes. Lets see metadata_grade class. It has 4 sub-classes: elementary school, high school, college, and daily life.

We have to experimentally tune some parameters. We plot all our points (5000), set perplexity to 2000, and set n_iter to 5000. These have given the best results. The results are shown below.



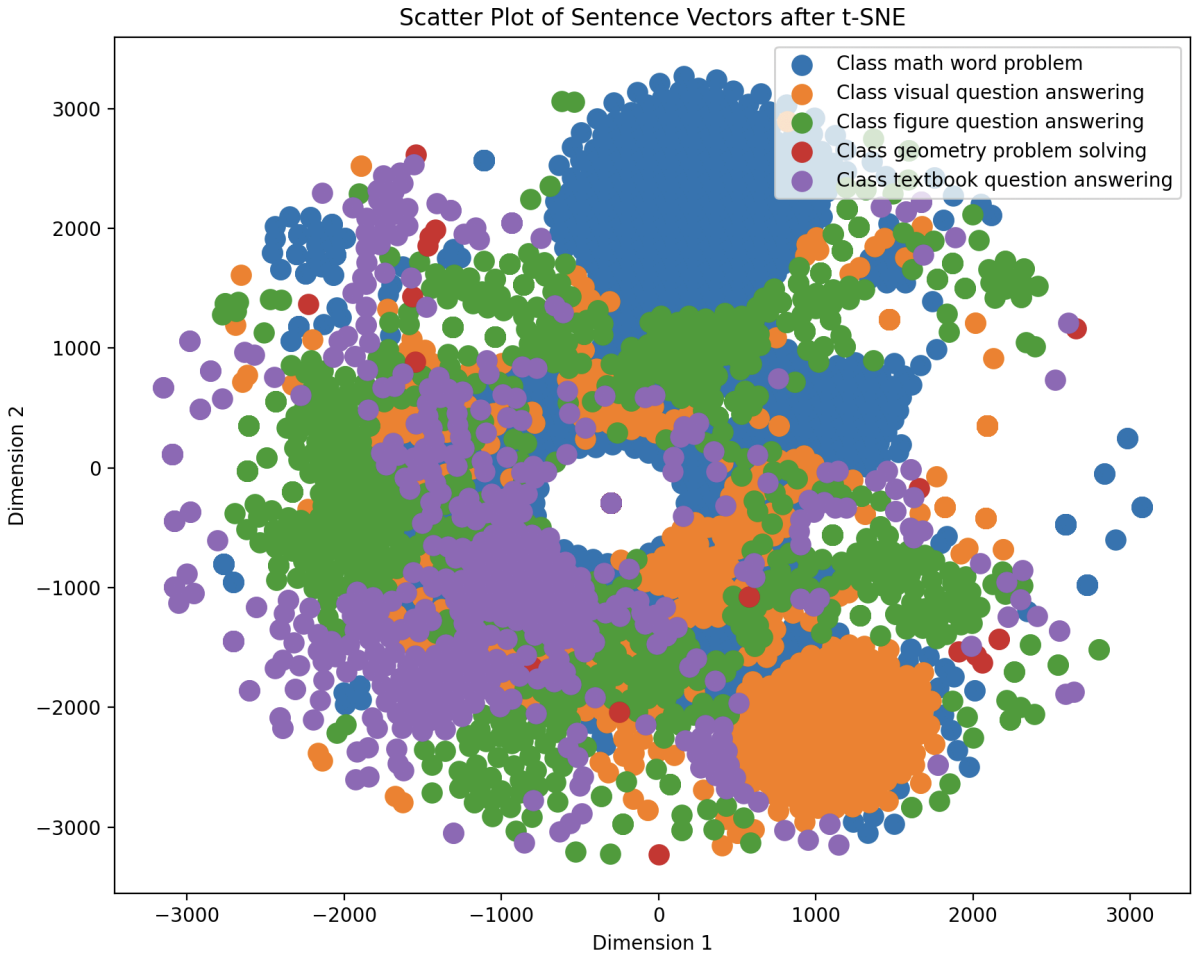
We see that the different clusters are quite prominent. Some points might look out of cluster, but it looks like that because of heavy compression in dimension.

We will also try to see some other classes. Let's see the class ‘answer type’:



We can see that the text class and integer class are quite different, the float mixes with text, but this can be because the number of points in a float is very low compared to text.

We will now try to see the “metadata task” class. These classes are well separated, we have set perplexity to 1000 and iteration to 10,000.



Result :

- A secure API is created.
- We have explained which words in a prompt have more influence than the others.
- We have seen the seperability of classes after compressing it to 2 dimensions.