Tanmay Suhas Jagtap

Portfolio : https://78-t0b1.github.io/Portfolio.github.io/

Linkein : https://www.linkedin.com/in/tanmay-jagtap-t0b1/

Github : https://github.com/78-t0b1

Week 2

Text Representation

Lexical Semantics

Some words have similar meanings (cat is similar to dog), others are antonyms (cold is the opposite of hot), some have positive connotations (happy) while others have negative connotations (sad). It should represent the fact that the meanings of buy, sell, and pay offer differing perspectives on the same underlying purchasing event. (If I buy something from you, you've probably sold it to me, and I likely paid you.) More generally, a model of word meaning should allow us to draw inferences to address meaning-related tasks like question-answering or dialogue.

Lemmas and Senses: Let's start by looking at how one word (we'll choose mouse) might be defined in a dictionary (simplified from the online dictionary WordNet): mouse (N)

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

lemma Here the form mouse is the lemma, also called the citation form. The form citation form mouse would also be the lemma for the word mice; dictionaries don't have separate definitions for inflected forms like mice. Similarly sing is the lemma for sing, sang, sung. In many languages the infinitive form is used as the lemma for the verb, so Spanish dormir "to sleep" is the lemma for duermes "you sleep". The specific forms wordform sung or carpets or sing or duermes are called wordforms. As the example above shows, each lemma can have multiple meanings; the lemma mouse can refer to the rodent or the cursor control device. We call each of these aspects of the meaning of mouse a word sense. The fact that lemmas can be polysemous (have multiple senses) can make interpretation difficult (is someone who types "mouse info" into a search engine looking for a pet or a tool?). Chapter 11 and Chapter 23 will discuss the problem of polysemy, and introduce word sense disambiguation, the task of determining which sense of a word is being used in a particular context.

Synonymy: A more formal definition of synonymy (between words rather than senses) is that two words are synonymous if they are substitutable for one another in any sentence

without changing the truth conditions of the sentence, the situations in which the sentence would be true. While substitutions between some pairs of words like car / automobile or water / H2O are truth preserving, the words are still not identical in meaning. Indeed, probably no two words are absolutely identical in meaning.

Word Similarity: While words don't have many synonyms, most words do have lots of similar words. Cat is not a synonym of dog, but cats and dogs are certainly similar words. In moving from synonymy to similarity, it will be useful to shift from talking about relations between word senses (like synonymy) to relations between words (like similarity). The notion of word similarity is very useful in larger semantic tasks. Knowing how similar two words are can help in computing how similar the meaning of two phrases or sentences are, a very important component of tasks like question answering, paraphrasing, and summarization.

Word Relatedness: Consider the meanings of the words coffee and cup. Coffee is not similar to cup; they share practically no features (coffee is a plant or a beverage, while a cup is a manufactured object with a particular shape). But coffee and cup are clearly related; they are associated by co-participating in an everyday event (the event of drinking coffee out of a cup). Similarly scalpel and surgeon are not similar but are related eventively (a surgeon tends to make use of a scalpel). One common kind of relatedness between words is if they belong to the same **semantic field**. A semantic field is a set of words which cover a particular semantic domain and bear structured relations with each other. Semantic fields are also related to topic models, like Latent Dirichlet Allocation, LDA, which apply unsupervised learning on large sets of texts to induce sets of associated words from text. Semantic fields and topic models are very useful tools for discovering topical structure in documents.

Semantic Frames and Roles: A semantic frame is a set of words that denote perspectives or participants in a particular type of event. A commercial transaction, for example, is a kind of event in which one entity trades money to another entity in return for some good or service, after which the good changes hands or perhaps the service is performed. This event can be encoded lexically by using verbs like buy (the event from the perspective of the buyer), sell (from the perspective of the seller), pay (focusing on the monetary aspect), or nouns like buyer. Frames have semantic roles (like buyer, seller, goods, money), and words in a sentence can take on these roles. Knowing that buy and sell have this relation makes it possible for a system to know that a sentence like Sam bought the book from Ling could be paraphrased as Ling sold the book to Sam, and that Sam has the role of the buyer in the frame and Ling the seller. Being able to recognize such paraphrases is important for question answering, and can help in shifting perspective for machine translation.

Connotation: The word connotation has different meanings in different fields, but here we use it to mean the aspects of a word's meaning that are related to a writer or reader's emotions, sentiment, opinions, or evaluations. For example some words have positive connotations(wonderful) while others have negative connotations (dreary). Even words whose meanings are similar in other ways can vary in connotation; consider the difference in connotations between fake, knockoff, forgery, on the one hand, and copy,

replica, reproduction on the other, or innocent (positive connotation) and naive (negative connotation). Some words describe positive evaluation (great, love) and others negative evaluation (terrible, hate). Positive or negative evaluation language is called sentiment.

Vector Semantics

Vector semantics is the standard way to represent word meaning in NLP, helping us model many of the aspects of word meaning.

Their idea was that two words that occur in very similar distributions (whose neighboring words are similar) have similar meanings.

For example, suppose you didn't know the meaning of the word ongchoi (a recent borrowing from Cantonese) but you see it in the following contexts:

(6.1) Ongchoi is delicious sauteed with garlic.

(6.2) Ongchoi is superb over rice.

(6.3) ...ongchoi leaves with salty sauces...

And suppose that you had seen many of these context words in other contexts:

(6.4) ...spinach sauteed with garlic over rice...

(6.5) ...chard stems and leaves are delicious...

(6.6) ...collard greens and other salty leafy greens

The fact that ongchoi occurs with words like rice and garlic and delicious and salty, as do words like spinach, chard, and collard greens might suggest that ongchoi is a leafy green similar to these other leafy greens. We can do the same thing computationally by just counting words in the context of ongchoi.

The idea of vector semantics is to represent a word as a point in a multidimensional semantic space that is derived from the distributions of word neighbors. Vectors for representing words are called embeddings.

Words and Vectors

Vector or distributional models of meaning are generally based on a co-occurrence matrix, a way of representing how often words co-occur.

Vectors and documents

In a term-document matrix, each row represents a word in the vocabulary and each column represents a document from some collection of documents.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Table shows a small selection from a term-document matrix showing the occurrence of four words in four plays by Shakespeare. Each cell in this matrix represents the number of times a particular word (defined by the row) occurs in a particular document (defined by the column). Thus fool appeared 58 times in Twelfth Night.

So As You Like It is represented as the list [1,114,36,20] (the first column vector in Fig. 6.3) and Julius Caesar is represented as the list [7,62,1,2] (the third column vector).

Term-document matrices were originally defined as a means of finding similar documents for the task of document information retrieval. Two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar.

Words as vectors: document dimensions

We've seen that documents can be represented as vectors in a vector space. But vector semantics can also be used to represent the meaning of words. We do this row vector by associating each word with a word vector— a row vector rather than a column vector.

The four dimensions of the vector for fool, [36,58,1,4], correspond to the four Shakespeare plays. Word counts in the same four dimensions are used to form the vectors for the other 3 words: wit, [20,15,2,3]; battle, [1,0,7,13]; and good [114,80,62,89].

For documents, we saw that similar documents had similar vectors, because similar documents tend to have similar words. This same principle applies to words: similar words have similar vectors because they tend to occur in similar documents. The term-document matrix thus lets us represent the meaning of a word by the documents it tends to occur in.

Words as vectors: word dimensions

An alternative to using the term-document matrix to represent words as vectors of document counts, is to use the term-term matrix, also called the word-word matrix or the term-context matrix, in which the columns are labeled by words rather than documents.

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Cosine for measuring similarity

To measure similarity between two target words v and w, we need a metric that takes two vectors (of the same dimensionality, either both with words as dimensions, hence of length |V|, or both with documents as dimensions, of length |D|) and gives a measure of their similarity. By far the most common similarity metric is the cosine of the angle between the vectors.

cosine(v,w) = v ·w /|v||w|

TF-IDF

https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

Term Frequency: In document d, the frequency represents the number of instances of a given word t. Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational. Since the ordering of terms is not significant, we can use a vector to describe the text in the bag of term models. For each specific term in the paper, there is an entry with the value being the term frequency.

tf(t,d) = count of t in d / number of words in d

Document Frequency: This tests the meaning of the text, which is very similar to TF, in the whole corpus collection. The only difference is that in document d, TF is the frequency counter for a term t, while df is the number of occurrences in the document set N of the term t. In other words, the number of papers in which the word is present is DF.

df(t) = occurrence of t in documents

Inverse Document Frequency: Mainly, it tests how relevant the word is. The key aim of the search is to locate the appropriate records that fit the demand. Since tf considers all terms equally significant, it is therefore not only possible to use the term frequencies to measure the weight of the term in the paper. First, find the document frequency of a term t by counting the number of documents containing the term:

df(t) = N(t)

where

df(t) = Document frequency of a term t

N(t) = Number of documents containing the term t

idf(t) = N/ df(t) = N/N(t)

The more common word is supposed to be considered less significant, but the element (most definite integers) seems too harsh. We then take the logarithm (with base 2) of the inverse frequency of the paper. So the if of the term t becomes:

idf(t) = log(N/ df(t))

tf-idf(t, d) = tf(t, d) * idf(t)

In [ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

d0 = 'Geeks for geeks'
d1 = 'Geeks'
d2 = 'r2j'

# merge documents into a single corpus
string = [d0, d1, d2]

# create object
tfidf = TfidfVectorizer()

# get tf-df values
result = tfidf.fit_transform(string)

# get idf values
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names_out(), tfidf.idf_):
        print(ele1, ':', ele2)
```

```
idf values:
for : 1.6931471805599454
geeks : 1.2876820724517808
r2j : 1.6931471805599454
```

In [ ]:
```
# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

# in matrix form
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

```
Word indexes:
{'geeks': 1, 'for': 0, 'r2j': 2}

tf-idf value:
  (0, 0)        0.5493512310263033
  (0, 1)        0.8355915419449176
  (1, 1)        1.0
  (2, 2)        1.0

tf-idf values in matrix form:
[[0.54935123 0.83559154 0.        ]
 [0.         1.         0.        ]
 [0.         0.         1.        ]]
```

What is Word Embedding?

The initial embedding techniques dealt with only words. Given a set of words, you would generate an embedding for each word in the set. The simplest method was to one-hot encode the sequence of words provided so that each word was represented by 1 and other words by 0. While this was effective in representing words and other simple text-processing tasks, it didn't really work on the more complex ones, such as finding similar words.

For example, if we search for a query: Best Italian restaurant in Delhi, we would like to get search results corresponding to Italian food, restaurants in Delhi and best. However, if we get a result saying: Top Italian food in Delhi, our simple method would fail to detect the similarity between 'Best' and 'Top' or between 'food' and 'restaurant'.

This issue gave rise to what we now call word embeddings. Basically, a word embedding not only converts the word but also identifies the semantics and syntaxes of the word to build a vector representation of this information. Some popular word embedding techniques include Word2Vec, GloVe, ELMo, FastText, etc.

The underlying concept is to use information from the words adjacent to the word. There have been path-breaking innovation in Word Embedding techniques with researchers finding better ways to represent more and more information on the words, and possibly scaling these to not only represent words but entire sentences and paragraphs.

In [ ]:
```python
import spacy

# Load the pre-trained model
nlp = spacy.load("en_core_web_sm")

# Example words
words = ["dog", "cat", "banana"]

# Get vectors for individual words
for word in words:
    vector = nlp(word).vector
    print(f"Vector for '{word}': {vector}")
```

```
Vector for 'dog': [-1.6806675  -1.2663747  -0.71255565  0.22143888  0.28581634
0.23924345
  1.2992647   1.0683641  -0.1666507  -0.593021    0.20207635 -0.71124184
 -0.5710875  -0.2685267  -0.5052826   0.60505986 -1.5851773  -1.6874862
  0.7026561   0.60366225  0.3043416   1.3963956  -0.056483   -0.6299367
  0.09717859  0.5463655   0.36506647  0.73901325 -0.16906115  0.35410628
  0.33770823 -0.7352046   1.6755302   0.48371333  0.0184648  -0.92315257
  0.6245377   0.11393103  0.8193037  -0.01115507 -0.49064368 -0.30198318
  0.43095675 -0.05127436 -0.11000359 -0.64060974 -1.4619632   0.85834503
 -0.4855454   0.01614086 -0.10124743  1.2471893   0.7936216  -0.49573362
  1.0994403  -0.22723481  1.289906   -0.8966851  -0.07580797 -0.6877714
 -0.9954758  -0.70957065 -0.3484952  -0.35015944  0.6045856   0.21346438
 -0.22741812  0.12088367  0.8521288   0.11694434  0.42555034  0.8777968
  1.9081106  -0.62818205  0.2991566  -0.26263964 -0.4627701  -0.11653326
  0.6030469  -1.1594303  -0.3226232  -0.1124312  -1.5564214  -0.44001883
  0.8246197   1.0218511   0.31111258 -0.46998724 -2.129951    1.1728595
 -1.2842656  -1.1179041   1.596463    0.51193094  0.22032768  1.6200272 ]
Vector for 'cat': [-1.3749216  -0.9209707  -0.48585108  0.7449953  -0.04045266
0.40948564
  1.2951561   1.480995    0.48480064 -0.7330386   2.462772   -0.9649264
 -0.37727293  0.08553091 -0.24743515  1.6335745  -0.06138834 -1.5028492
  0.967414    0.63307035 -0.72185993  0.6233178  -1.1602921  -0.6722332
  0.9362452   1.0436772   0.3563193   0.9629261  -0.8357587   0.449831
 -0.6676395  -0.31429762  1.9747405  -0.38430327 -0.44635606 -2.0205777
  0.07270992  1.4437965   0.5954057   0.31014535 -1.2809043  -0.60996306
 -0.6490139   0.72085637 -1.1413931  -1.5718629  -0.5458978   1.5909092
  0.38662726  0.47634625  0.70607305  0.91097486  0.34076473  0.32807004
 -0.34836677  0.07606278  1.0374578   0.24974427  0.21037751 -0.10116711
 -0.97328275 -0.95685375 -0.7718992  -0.6154634  -0.19992816 -0.19895095
 -0.4496122   0.28037518  0.5823548  -1.214601    1.1826286   0.5630324
  0.1241945  -1.0164824   0.14934509 -0.3980039  -0.9265938   0.21561976
  0.39138982 -2.1365113  -0.29684037 -0.23177594 -1.2740371  -1.3451545
  0.5225169   0.99557483 -0.05220089 -0.0129011  -1.2571102   1.2818379
 -0.379385    0.57722074  1.571756    0.54320335 -0.07744253  2.266047  ]
Vector for 'banana': [-1.43172    -1.0467288  -0.780927    0.45387337  0.22313687
-0.4822587
  0.7796049   1.9291902   0.5263105  -0.8860629   0.31216276 -0.8671374
 -0.97779465  0.62443805 -0.24205405  0.70003366 -1.1135058  -1.335113
  1.551111   -0.27649075 -0.49291223  0.21266821 -1.2886872  -0.4015749
  1.4845917   0.26011398  0.244566    1.1891397  -0.4271981   0.32347038
 -0.2853762   0.9585209   1.3764503   0.02244225 -0.4406012  -2.0229402
  0.14719443  1.6322699   0.8777428  -1.1011673  -0.5580491   0.16557175
 -0.49441516  0.7951485  -0.7971488  -0.04855184 -0.6682936   0.65149677
  0.28960374  0.02061486  0.97740465  0.2379916   0.72662365 -0.71118784
  0.10751665 -0.0982407   0.66061974 -0.23507589  0.304424   -0.65600216
 -1.2755834  -0.41828483  1.0467389  -0.20455568  0.1618619   0.6822717
 -0.06672494 -0.16852754  0.12145358 -0.42135045 -0.13758652  1.6141541
  0.9650748   0.63999003 -0.14702561  0.24284407  0.14128391 -0.6678508
 -0.32493967 -1.2624674   0.2822321  -0.96413857 -0.6104078  -0.4091128
 -0.18481803  1.1050515  -0.55165374  0.17582801 -1.0846602   0.35955024
 -1.5655494  -0.54753345  2.4091485   0.516353    0.55651164  1.1028976 ]
```

In [ ]:
```python
# Example sentence
sentence = "The quick brown fox jumps over the lazy dog."

# Get the vector for the entire sentence
sentence_vector = nlp(sentence).vector
print(f"Vector for the sentence: {sentence_vector}")
```

```
Vector for the sentence: [ 0.16722472 -0.41780248  0.21882531  0.38652307 -0.2085
5291  0.1814196
  0.2028152  -0.2612236   0.13432345  0.10539548  0.47338143  0.26623455
 -0.2694387  -0.2744369  -0.20732066  0.1264784  -0.28658912  0.25923425
 -0.13064954 -0.08419564 -0.35259548  0.03415954 -0.12539047 -0.3591571
  0.16366193  0.19067621  0.2338213   0.14691558  0.45343202  0.43327522
  0.6339363  -0.32620642  0.28089175 -0.31701192  0.07674997  0.39677358
  0.52545494 -0.08408366 -0.21826155  0.23854144 -0.21939552  0.5288199
 -0.11836058  0.01887446 -0.02045707 -0.1735549  -0.11093439  0.09989184
 -0.01064302  0.19542125 -0.35433078  0.9176275  -0.25008193  0.0549965
 -0.07219964  0.1983386  -0.15006927  0.34726363 -0.31189656 -0.10487971
 -0.40145794 -0.56066954 -0.06650138 -0.05759991  0.0078478  -0.04010873
  0.23358583 -0.00149248 -0.0379594  -0.36177197  0.35952622  0.35372975
  0.12966701 -0.22811572  0.07749183 -0.7927288  -0.21754682 -0.5236887
 -0.09702981 -0.7740661  -0.23893821  0.23108883 -0.6463502  -0.25897202
 -0.08455285  0.16531123 -0.01802757 -0.01556575 -0.62966764  0.47165197
  0.03968055  0.20545992  0.69730294  0.33681795 -0.4277688   0.02319126]
```

In [ ]:

```python
# Example words
word1 = nlp("dog")
word2 = nlp("cat")
word3 = nlp("banana")

# Calculate similarity
similarity_1_2 = word1.similarity(word2)
similarity_1_3 = word1.similarity(word3)

print(f"Similarity between 'dog' and 'cat': {similarity_1_2}")
print(f"Similarity between 'dog' and 'banana': {similarity_1_3}")

# Example sentences
sentence1 = nlp("The quick brown fox jumps over the lazy dog.")
sentence2 = nlp("A fast, brown fox leaps over a sleepy dog.")
sentence3 = nlp("A banana is yellow.")

# Calculate similarity
sentence_similarity_1_2 = sentence1.similarity(sentence2)
sentence_similarity_1_3 = sentence1.similarity(sentence3)

print(f"Similarity between sentence 1 and sentence 2: {sentence_similarity_1_2}"
print(f"Similarity between sentence 1 and sentence 3: {sentence_similarity_1_3}"
```

```
Similarity between 'dog' and 'cat': 0.6847176149951816
Similarity between 'dog' and 'banana': 0.6648012515757744
Similarity between sentence 1 and sentence 2: 0.7417474972394386
Similarity between sentence 1 and sentence 3: 0.3030585350922879
```

```
C:\Users\tanma\AppData\Local\Temp\ipykernel_284\1554730959.py:7: UserWarning: [W0
07] The model you're using has no word vectors loaded, so the result of the Doc.s
imilarity method will be based on the tagger, parser and NER, which may not give
useful similarity judgements. This may happen if you're using one of the small mo
dels, e.g. `en_core_web_sm`, which don't ship with word vectors and only use cont
ext-sensitive tensors. You can always add your own word vectors, or use one of th
e larger models instead if available.
  similarity_1_2 = word1.similarity(word2)
C:\Users\tanma\AppData\Local\Temp\ipykernel_284\1554730959.py:8: UserWarning: [W0
07] The model you're using has no word vectors loaded, so the result of the Doc.s
imilarity method will be based on the tagger, parser and NER, which may not give
useful similarity judgements. This may happen if you're using one of the small mo
dels, e.g. `en_core_web_sm`, which don't ship with word vectors and only use cont
ext-sensitive tensors. You can always add your own word vectors, or use one of th
e larger models instead if available.
  similarity_1_3 = word1.similarity(word3)
C:\Users\tanma\AppData\Local\Temp\ipykernel_284\1554730959.py:19: UserWarning: [W
007] The model you're using has no word vectors loaded, so the result of the Doc.
similarity method will be based on the tagger, parser and NER, which may not give
useful similarity judgements. This may happen if you're using one of the small mo
dels, e.g. `en_core_web_sm`, which don't ship with word vectors and only use cont
ext-sensitive tensors. You can always add your own word vectors, or use one of th
e larger models instead if available.
  sentence_similarity_1_2 = sentence1.similarity(sentence2)
C:\Users\tanma\AppData\Local\Temp\ipykernel_284\1554730959.py:20: UserWarning: [W
007] The model you're using has no word vectors loaded, so the result of the Doc.
similarity method will be based on the tagger, parser and NER, which may not give
useful similarity judgements. This may happen if you're using one of the small mo
dels, e.g. `en_core_web_sm`, which don't ship with word vectors and only use cont
ext-sensitive tensors. You can always add your own word vectors, or use one of th
e larger models instead if available.
  sentence_similarity_1_3 = sentence1.similarity(sentence3)
```

What is Sentence Embedding?

In NLP, sentence embedding refers to a numeric representation of a sentence in the form of a vector of real numbers, which encodes meaningful semantic information. It enables comparisons of sentence similarity by measuring the distance or similarity between these vectors. Techniques like Universal Sentence Encoder (USE) use deep learning models trained on large corpora to generate these embeddings, which find applications in tasks like text classification, clustering, and similarity matching.

Suppose, we come across a sentence like ‘I don’t like crowded places’, and a few sentences later, we read ‘However, I like one of the world’s busiest cities, New York’. How can we make the machine draw the inference between ‘crowded places’ and ‘busy cities’?

Clearly, word embedding would fall short here, and thus, we use Sentence Embedding. Sentence embedding techniques represent entire sentences and their semantic information as vectors. This helps the machine in understanding the context, intention, and other nuances in the entire text.

Methods of Sentence Embedding

Several methods are employed to generate sentence embeddings:

Averaging Word Embeddings: This approach involves taking the average of word embeddings within a sentence. While simple, it may not capture complex contextual nuances.

Pre-trained Models like BERT: Models like BERT (Bidirectional Encoder Representations from Transformers) have revolutionized sentence embeddings. BERT-based models consider the context of each word in a sentence, resulting in rich and contextually aware embeddings.

Neural Network-Based Approaches: Skip-Thought vectors and InferSent are examples of neural network-based sentence embedding models. They are trained to predict the surrounding sentences, which encourages them to understand sentence semantics.

Noteworthy Sentence Embedding Models

BERT (Bidirectional Encoder Representations from Transformers): BERT has set a benchmark in sentence embeddings, offering pre-trained models for various NLP tasks. Its bidirectional attention and contextual understanding make it a prominent choice.

RoBERTa: An evolution of BERT, RoBERTa fine-tunes its training methodology, achieving state-of-the-art performance in multiple NLP tasks.

USE (Universal Sentence Encoder): Developed by Google, USE generates embeddings for text that can be used for various applications, including cross-lingual tasks.

In [ ]:
```
from transformers import BertModel, BertTokenizer
import torch

# Load pre-trained BERT model and tokenizer
model_name = 'bert-base-uncased'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertModel.from_pretrained(model_name)

# Function to get sentence embedding
def get_sentence_embedding(sentence):
    # Tokenize input sentence and get token IDs
    inputs = tokenizer(sentence, return_tensors='pt', truncation=True, padding=T

    # Get the hidden states from the last layer
    with torch.no_grad():
        outputs = model(**inputs)
        hidden_states = outputs.last_hidden_state

    # Average pooling of the token embeddings to get the sentence embedding
    sentence_embedding = hidden_states.mean(dim=1)
    return sentence_embedding

# Example sentences
sentences = [
    "The quick brown fox jumps over the lazy dog.",
    "A fast, brown fox leaps over a sleepy dog.",
    "A banana is yellow."
]
```

```python
# Get embeddings for each sentence
embeddings = [get_sentence_embedding(sentence) for sentence in sentences]

for i, embedding in enumerate(embeddings):
    print(f"Embedding for sentence {i+1}: {embedding}")
```

```
Downloading vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
c:\Users\tanma\.conda\envs\LLM_implmnt\lib\site-packages\huggingface_hub\file_download.py:133: UserWarning: `huggingface_hub` cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\tanma\.cache\huggingface\hub. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
Downloading tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
Downloading config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
Downloading model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]
```

```
Embedding for sentence 1: tensor([[-1.4466e-02, -7.4888e-02,  5.6368e-02,  4.5166
e-03,  4.0891e-01,
          2.5804e-02, -7.5612e-02,  4.7453e-01, -1.8954e-03, -1.5011e-01,
         -1.0119e-01, -1.5718e-01, -2.1705e-01, -1.6252e-02, -4.5524e-01,
         -2.5191e-01,  2.0256e-01, -2.0102e-02, -1.6217e-01, -8.5939e-03,
          1.9837e-01, -3.7650e-01, -5.1490e-01, -6.7321e-02,  4.7553e-01,
          2.2702e-01, -3.7371e-03,  2.4479e-01, -3.7154e-01,  1.7458e-02,
          2.2193e-01, -1.3309e-01, -1.1020e-02,  1.4926e-01, -1.6918e-01,
         -3.3690e-02,  4.0328e-02, -3.5493e-01, -4.4815e-01,  8.7867e-02,
         -2.4535e-01, -5.4502e-02, -8.5849e-02, -8.2122e-02,  1.0064e-01,
         -4.1189e-01,  6.9579e-02, -2.2557e-01,  7.4127e-01, -3.2274e-01,
         -5.2064e-01,  5.7741e-01, -3.2543e-01,  1.8200e-01, -3.3087e-01,
          2.8472e-01,  3.9729e-01, -1.5372e-01,  1.5706e-01, -2.7112e-01,
         -1.5829e-01,  3.0283e-01,  3.3324e-01, -1.7339e-01, -3.6901e-01,
          4.4814e-01, -2.4117e-01,  3.0746e-01, -1.8013e-01,  1.7885e-01,
          7.5899e-03, -3.9233e-01, -1.2223e-01,  3.3068e-01, -2.6184e-02,
          2.7726e-01,  1.9566e-02,  3.7135e-01, -3.5426e-02,  2.1945e-01,
         -2.1727e-02,  3.4998e-01,  6.2089e-02,  6.4642e-01,  1.4670e-01,
          9.6474e-02, -4.7752e-02, -5.6338e-04, -8.1221e-01,  5.2900e-01,
         -2.0227e-01, -8.7444e-02, -4.4891e-01,  2.8807e-01, -2.2788e-01,
          8.5220e-02,  1.5184e-01, -4.2879e-01,  3.0909e-01,  1.6595e-01,
          1.1818e-01, -1.0160e-01, -1.9517e-01,  3.3915e-01, -4.1961e-01,
         -3.1387e-01,  2.1041e-01,  1.1116e-01, -4.6362e-01,  2.0969e-01,
          2.5672e-02, -3.0196e-01,  2.8832e-01, -7.1632e-02,  1.0036e-01,
          6.0099e-01, -5.3618e-02,  5.8555e-03, -4.0532e-01, -5.7700e-02,
          1.0334e-01, -1.6271e-01, -5.4231e-02,  6.0282e-01,  1.1722e-01,
         -1.2507e-02, -2.3295e-01,  2.8025e-01,  3.0765e-01, -4.8797e-01,
          3.7415e-01,  6.4034e-01,  1.1006e-02, -7.2071e-01, -1.2806e-01,
          7.0593e-01,  3.5190e-01, -4.0405e-01, -4.7778e-01, -1.5564e-01,
          9.7278e-02, -6.2522e-02,  3.9642e-01,  5.4876e-02,  5.1176e-01,
          5.2046e-02,  6.1899e-02, -2.6804e-01,  7.2896e-02,  2.7116e-01,
          5.8959e-02, -1.7484e-02, -2.6138e-01,  2.6955e-02, -3.4479e-01,
         -2.7034e-03, -1.9064e-01,  2.2388e-01,  3.1121e-01,  7.0139e-01,
          4.5558e-01, -2.9986e-01, -3.8799e-02, -3.5566e-01, -2.2397e-01,
          2.1785e-01, -1.2092e-02,  3.5232e-01, -7.6077e-01, -2.5159e-02,
         -1.9849e-01,  2.5566e-01,  8.1111e-01,  3.9301e-02,  6.4932e-02,
          1.6473e-03, -9.1074e-02, -3.0760e-01,  8.6632e-02, -4.6544e-02,
         -1.6814e+00, -3.1919e-02,  3.9190e-01,  2.5601e-01,  2.6399e-01,
         -7.0980e-01,  2.8260e-01, -4.5276e-01,  8.8361e-02, -4.9906e-01,
          6.9349e-03, -1.9329e-01, -3.7027e-01, -1.8294e-01,  2.2988e-01,
         -7.0672e-01, -2.8422e-01, -1.0568e+00, -6.9574e-02, -2.3391e-01,
         -1.9285e-04, -2.3800e-01,  2.4757e-01,  1.4585e-01, -1.3435e-01,
          2.6265e-01, -3.9125e-02, -1.2208e-01, -9.6399e-02,  2.2775e-01,
         -4.1947e-01,  4.8615e-01,  4.5283e-02,  3.0925e-01,  1.5423e-01,
         -6.2226e-02, -1.3138e-02, -1.0831e-01, -9.0518e-02,  4.4261e-01,
         -1.3887e-01,  2.5485e-02, -7.7789e-01,  3.4352e-01, -9.0903e-02,
          6.1130e-01,  3.3316e-01, -3.0117e-01, -7.8062e-03,  2.6245e-01,
          3.8898e-02, -4.4499e-01,  2.7017e-01,  2.9772e-01, -4.8468e-01,
         -3.7126e-01, -3.0810e-01, -1.9069e-01,  3.7637e-01, -2.1820e-01,
         -6.8321e-02,  4.7604e-01,  7.2020e-01, -1.3729e-01, -1.6398e-02,
         -2.3122e-01,  2.8808e-01,  2.9886e-01, -6.3861e-01, -6.5560e-01,
          2.0686e-01, -6.2668e-01,  5.8434e-01, -6.7680e-01, -1.7181e-02,
         -4.6208e-01, -9.1070e-02,  2.5591e-01,  1.1651e-01,  8.8600e-02,
          6.1591e-01, -2.5783e-01,  4.0150e-01,  1.7519e-01, -5.7760e-02,
         -3.1138e-01, -5.4169e-02,  3.8096e-01, -1.4464e-01, -1.1587e-01,
         -9.4042e-02, -2.3702e-01,  6.1146e-01,  8.7330e-02, -3.4574e-01,
         -4.6055e-01,  5.5924e-01, -1.7001e-01,  1.9079e-01, -5.7535e-01,
          3.3122e-01,  3.8111e-01, -6.4289e-01,  1.9750e-01,  2.4475e-01,
         -3.1818e-01,  1.1561e-01, -2.7848e-01, -3.5097e-01, -3.9067e-01,
         -9.1297e-02,  5.8796e-01, -1.3698e-01, -1.2536e-01, -5.9157e-01,
```

```
 5.2725e-02,  9.3350e-02,  3.2943e-01,  4.6846e-01, -2.8123e-01,
 2.2552e-01,  1.3681e-01,  5.4509e-02,  5.2443e-02, -1.5735e-02,
 2.2778e-03, -7.1222e-03, -2.4675e-02, -3.8425e+00, -1.6402e-02,
 1.9749e-01, -2.2693e-01,  4.0033e-01, -4.6595e-01, -3.1480e-01,
-2.4757e-01, -8.0993e-01,  5.4506e-01, -3.2937e-02, -4.7280e-01,
 5.7032e-01,  1.1619e-01,  5.7917e-01, -4.9033e-01,  3.5363e-02,
-2.4676e-01, -5.1750e-01,  4.6270e-01, -2.7722e-01, -5.3776e-01,
-4.0643e-02, -3.9667e-01,  2.5786e-01,  3.1580e-01, -8.1626e-02,
-4.0290e-01, -2.2949e-01, -2.1664e-01,  1.3059e-01, -3.3093e-01,
 3.5602e-01,  1.6516e-01, -5.1379e-02,  1.3124e-01,  6.8181e-02,
 5.6517e-02,  1.6310e-01,  2.5229e-01,  2.6072e-01, -4.7854e-02,
 8.5442e-02,  1.8037e-01,  5.4632e-01, -8.9347e-02,  4.4117e-02,
-6.6599e-01,  4.3147e-01,  2.1994e-01, -2.5459e-02, -8.8054e-03,
-2.7057e-01, -2.2525e-02, -1.0167e-01, -2.6819e-01,  2.1538e-01,
 4.3078e-01, -4.8317e-01, -3.6039e-01,  2.4065e-01, -4.2511e-01,
-3.9692e-01,  4.2681e-01,  3.6141e-01, -5.4780e-01, -4.1091e-01,
-2.7886e-01, -1.2552e-01,  4.3613e-01,  1.7238e-01,  1.5567e-02,
-3.0039e-01, -1.2287e+00, -1.1025e-02,  6.0471e-02,  2.9954e-01,
-3.6389e-01, -3.6131e-02, -3.9911e-01,  5.3104e-02, -4.8018e-01,
-1.1655e-01,  1.5605e-02, -8.5507e-02, -8.6772e-01, -1.6309e-01,
-1.1853e-01, -1.8841e-01, -4.6579e-01,  2.1308e-01,  2.8534e-01,
-2.1620e-01,  3.9187e-01,  3.6541e-02, -1.5135e-02,  2.2076e-01,
-2.7811e-01,  1.3092e-01, -3.1637e-01,  2.3382e-01, -1.0502e-01,
 2.8414e-01,  2.4795e-01,  4.1124e-02,  1.0705e-01, -4.7802e-01,
 3.2240e-02,  5.5115e-02,  1.7229e-01,  2.7635e-02, -7.3531e-01,
 3.7128e-01, -3.6728e-01, -2.2378e-01,  1.5550e-01,  7.3016e-01,
 4.5918e-01, -7.7650e-02,  3.3790e-01, -3.4099e-01,  5.0022e-01,
-1.8387e-01, -3.3796e-01,  1.1402e-01, -7.3288e-03, -5.9819e-02,
-2.3557e-01, -3.9225e-02, -3.0966e-01, -1.5522e-01, -1.6682e-01,
 4.9195e-02, -4.0943e-02,  1.0525e-01,  4.3712e-01, -1.7919e-01,
-3.1464e-01,  2.0048e-01,  1.1368e-02, -9.0712e-02,  4.7407e-01,
 1.9130e-01,  7.2493e-03,  2.0193e-01,  2.4541e-01,  1.1609e-01,
 1.0626e-01,  1.3714e-01,  2.8068e-01, -2.0294e-01, -8.7983e-02,
 1.1514e-01, -1.4101e-01, -3.4613e-02, -3.2021e-01,  3.6586e-01,
-2.1094e-01,  1.5018e-03,  8.3327e-02, -1.3032e-02, -1.1281e-01,
 1.6577e-01,  4.8601e-02, -1.8862e-01,  5.3260e-01,  5.7785e-01,
-3.2149e-01, -2.8207e-01,  4.2305e-01,  9.4466e-02, -3.4878e-01,
 1.7316e-01,  2.6661e-01, -3.9486e-01,  3.3045e-01, -1.3838e-01,
-3.5525e-01, -4.1333e-01,  4.9462e-01, -1.0501e-01, -3.2042e-01,
-1.7482e-01, -7.4739e-02,  4.1072e-01, -3.3674e-02,  2.2061e-01,
-4.8677e-02,  4.4808e-01,  1.4515e-01, -6.6817e-02,  7.8852e-01,
-1.9754e-02,  1.9737e-01, -7.7491e-01, -3.6592e-01,  3.8936e-02,
 4.4102e-01,  1.7138e-01, -1.4712e-01, -4.1784e-02,  6.0054e-02,
 2.0394e-01, -2.4660e-01, -2.1239e-01, -1.3517e-01,  1.2439e-01,
 9.4923e-02, -6.3869e-03,  8.7993e-02, -4.1101e-01, -6.3185e-01,
-1.3541e-01, -3.6115e-01,  8.3716e-02,  4.7428e-01,  9.0565e-02,
-2.7417e-01, -3.0067e-01,  3.8710e-01, -2.4024e-01,  1.0256e-02,
-1.2699e-01, -4.7449e-02, -1.7271e-01, -1.1195e-01, -5.2120e-01,
 2.9314e-01, -1.2221e-01, -1.1372e-01,  5.6056e-01, -1.4113e-01,
-1.9770e-02, -4.6353e-02, -2.4437e-01,  2.1419e-01, -1.3879e-01,
-3.6466e-01,  3.2132e-01,  1.0389e-01,  4.5749e-01,  3.3237e-01,
 2.6618e-01, -3.3554e-02,  2.5043e-01, -1.5038e-01, -1.6469e-01,
 2.6327e-01, -2.2347e-01,  1.7536e-01,  4.1305e-01,  7.0896e-01,
 3.8733e-01, -1.1563e-01, -1.2500e-01, -3.6961e-02, -5.9878e-01,
-4.9016e-02,  4.0799e-01,  1.7423e-03, -1.5176e-01, -1.4090e-01,
-4.8255e-01,  5.0502e-01,  2.2385e-01, -5.8301e-02, -2.3691e-01,
 2.8384e-01,  3.0855e-01,  2.4265e-01,  2.6333e-02, -5.0807e-01,
 4.2046e-01, -3.7265e-01, -1.5975e-01,  2.0993e-01, -2.7342e-02,
-5.2103e-01, -2.3068e-01, -1.1104e-01, -1.4398e-02,  5.2140e-01,
-4.6261e-01,  1.5798e-01, -2.5219e-01, -2.3700e-01, -3.6891e-02,
```

```
          1.9005e-01, -1.0100e-01,  2.3024e-01,  9.2388e-02,  4.0274e-01,
          2.0482e-01, -1.6285e-01,  4.4482e-02,  1.6987e-01,  7.1627e-01,
          4.8801e-01, -1.1135e-01,  1.4667e-01, -5.4725e-01, -8.6478e-02,
         -6.9122e-02, -2.0371e-01, -1.6111e-01, -8.5644e-02, -1.0059e-02,
          2.2874e-01, -4.4259e-01,  3.5945e-01, -4.2526e-01, -3.0918e-01,
          1.4202e-01,  5.4228e-01, -3.3196e-01,  2.0588e-01, -5.2117e-01,
         -3.7526e-01,  2.7902e-02,  2.6390e-01,  1.0984e-01, -1.8775e-01,
          2.4814e-01,  1.2290e-01,  2.4260e-02,  2.6903e-01, -1.4120e-01,
          2.3879e-01,  1.8531e-02,  2.8040e-02, -1.5253e-02,  1.1651e-01,
          1.1166e-01,  3.4132e-01,  1.1574e-01, -3.4797e-01,  4.6232e-01,
          3.1002e-01, -5.0188e-02, -4.6480e-01,  6.7302e-02, -4.0957e-02,
         -6.0579e-02,  2.6706e-01,  1.5773e-01, -3.3525e-01,  2.7831e-01,
          5.7700e-02,  4.6349e-01,  4.0421e-01,  3.4571e-02,  2.9391e-01,
         -2.9531e-02, -2.0184e-01,  2.1205e-01, -1.0302e-01,  3.2663e-01,
          2.2959e-01, -2.4982e-01,  2.9891e-01,  3.2787e-01,  7.4868e-01,
         -3.7679e-03, -4.3431e-01,  8.9865e-02,  3.9354e-01,  1.2945e-01,
         -4.4494e-01,  4.6616e-02,  4.4272e-02,  4.3800e-01,  3.6034e-01,
         -2.2967e-01,  1.1138e-01,  1.8754e-01,  4.6500e-01, -1.6361e-01,
         -1.4334e-01, -4.3711e-01,  3.7022e-01, -2.8283e-01,  3.4558e-01,
         -5.2603e-01, -3.7587e-01,  3.3238e-02, -4.4326e-02, -3.2208e-01,
         -7.2395e-01, -1.4872e-01, -3.6032e-01, -1.7979e-01, -1.7507e-01,
          3.9200e-02, -2.9759e-01,  1.5413e-01, -3.7027e-01,  5.8414e-01,
          2.1532e-01,  2.4084e-01,  4.9964e-01,  5.7512e-01, -4.3322e-01,
         -1.4127e-01,  4.5748e-02,  5.2949e-01, -1.4470e-01,  1.3946e-01,
         -3.1020e-01, -1.3063e-01, -7.4903e-02,  3.9548e-01,  3.2388e-01,
         -3.4475e-01,  2.6025e-01, -8.5023e-02,  2.6431e-02,  1.8732e-01,
         -2.0028e-01, -4.5054e-01, -1.9796e-01,  1.1123e-01, -1.2287e-01,
          5.9471e-02, -4.2267e-02, -3.6902e-01, -3.1320e-02,  3.5532e-01,
          7.9503e-02, -4.0373e-02,  1.0681e-01, -2.0274e-01,  1.2537e-01,
          3.8011e-01, -2.8078e-01,  7.4515e-02, -1.2289e-01, -3.2250e-01,
         -5.0458e-02, -4.4619e-01, -6.6816e-01,  3.1076e-01,  2.9965e-01,
          3.2352e-01,  3.7372e-01, -9.8289e-03, -1.5577e-01,  8.3053e-02,
          2.0889e-01, -1.3343e-01, -5.6673e-01,  2.9133e-02,  5.8025e-02,
         -4.7680e-01,  3.4382e-02, -6.6228e-02, -2.6393e-01,  5.5996e-02,
         -2.6255e-01,  4.9537e-01,  7.3980e-02]])
Embedding for sentence 2: tensor([[ 1.4505e-01, -2.9002e-02, -1.3630e-01,  3.4652
e-02,  4.5365e-01,
         -1.2917e-02, -1.4032e-01,  3.0610e-01, -5.5195e-02, -2.2800e-01,
         -2.6457e-01, -1.9636e-01, -6.5769e-02,  7.0524e-02, -2.8842e-01,
          2.6904e-03,  2.0701e-01, -1.1788e-01,  2.1991e-02, -6.7864e-03,
          5.7309e-02, -2.4703e-01, -3.8139e-01, -1.3850e-02,  6.3807e-01,
          1.9246e-01,  1.0794e-02,  4.9500e-01, -2.9595e-01,  5.2878e-02,
          8.8554e-02,  2.2857e-01, -6.4616e-02,  1.6642e-01,  9.8124e-02,
          1.4027e-01,  2.7130e-01, -3.9195e-01, -3.6799e-01,  5.9372e-02,
         -3.1391e-01, -2.3018e-01, -2.7820e-01, -9.2925e-02,  1.3269e-01,
         -5.0740e-01,  1.7907e-01, -1.9792e-01,  6.0440e-01, -1.8859e-01,
         -4.3241e-01,  5.8770e-01, -4.3671e-01, -9.0682e-02, -2.4755e-01,
          4.1968e-01,  2.9356e-01, -2.8260e-01,  1.6767e-01,  1.9983e-02,
         -2.6296e-02, -3.1168e-02,  3.4052e-01, -3.4991e-01, -3.8327e-01,
          6.0059e-01, -4.5092e-01, -2.2428e-02, -1.6850e-01, -5.0647e-02,
         -4.6935e-02, -1.6889e-01, -9.7056e-02,  2.5081e-01, -1.0717e-01,
          2.5511e-01,  7.0924e-02,  3.1403e-01,  4.4098e-02,  2.2142e-01,
          1.6137e-01,  6.9821e-01,  2.3318e-01,  6.0353e-01,  2.6820e-01,
          7.6179e-02, -1.3296e-01,  1.6240e-01, -6.8740e-01,  3.5718e-01,
         -2.5103e-01, -3.0647e-01, -4.4499e-01,  5.3748e-01, -2.9999e-01,
          2.1789e-01,  1.2650e-01, -3.4885e-01,  1.9678e-01,  3.3903e-02,
          2.7993e-01, -2.7458e-01,  5.2070e-02,  3.6040e-01, -5.6672e-01,
         -1.2261e-01,  3.4546e-02,  9.3056e-02, -4.5899e-01,  2.6749e-01,
         -3.0587e-02, -3.8946e-01,  2.8089e-01,  8.3736e-02, -1.2342e-01,
          5.7151e-01,  9.3363e-02,  1.4724e-01, -4.6677e-01, -1.8464e-01,
```

```
 1.9414e-01, -3.5822e-01, -1.6564e-01,  4.5478e-01,  7.0507e-02,
-1.4961e-01, -1.5805e-01,  1.9274e-01,  3.0163e-01, -5.0067e-01,
 3.4994e-01,  3.8988e-01,  1.4948e-01, -7.0805e-01, -1.1081e-01,
 5.3116e-01,  2.2064e-01, -5.3330e-01, -5.9624e-01, -8.6926e-02,
 9.3850e-02, -1.9593e-02,  4.4927e-01,  4.4135e-02,  5.1389e-01,
 9.4337e-02,  2.9820e-01, -3.3477e-01,  3.0947e-02,  1.3571e-01,
 6.6815e-02,  9.0817e-02, -1.3608e-01,  1.9088e-01, -4.3829e-01,
 3.3862e-02, -1.0849e-01,  1.4031e-01,  4.3255e-01,  5.3591e-01,
 3.0215e-01, -2.4351e-01, -1.1329e-01, -3.8883e-01, -2.3992e-01,
 2.2118e-01,  7.1607e-03,  2.0093e-01, -9.1314e-01,  4.8621e-02,
-3.3714e-01,  3.7917e-01,  5.6149e-01,  6.0275e-03,  1.0146e-01,
-3.9890e-02,  1.1674e-01, -2.5564e-01,  1.1857e-01, -7.0551e-02,
-1.6968e+00, -2.2932e-01,  5.9520e-01,  2.4111e-01, -1.2572e-01,
-6.2358e-01,  3.3721e-02, -3.6732e-01,  3.8083e-02, -4.1251e-01,
-2.5059e-01,  3.3682e-02, -3.2642e-01,  1.1013e-01,  1.7525e-01,
-9.7225e-01, -1.3512e-01, -8.2536e-01, -2.9510e-01, -4.3408e-01,
-6.4662e-02, -2.8716e-01, -1.1722e-03,  2.5867e-01, -2.2731e-02,
 3.0669e-02,  1.0195e-01, -2.1339e-01, -2.9416e-01,  2.7821e-01,
-5.6061e-01,  3.4223e-01, -5.4004e-02,  5.0000e-01,  6.3113e-02,
 9.2377e-02, -9.5239e-02, -1.2570e-01, -2.3195e-01,  2.4218e-01,
-3.4019e-01, -1.9741e-02, -6.1159e-01,  3.4023e-01, -5.0206e-02,
 3.9090e-01,  4.4543e-01, -1.2976e-01, -6.8614e-02,  2.6929e-01,
-5.7419e-02, -3.7098e-01,  1.6157e-02,  1.4868e-01, -6.2596e-01,
-3.8333e-01, -2.9242e-01, -4.1906e-01,  4.0913e-01, -3.3397e-01,
-7.3317e-02,  1.8703e-01,  9.4921e-01,  5.4108e-03,  1.2556e-01,
-2.9979e-01,  1.4132e-01,  5.0223e-01, -4.1246e-01, -6.2708e-01,
-7.4374e-03, -6.1114e-01,  4.2845e-01, -4.3697e-01, -1.8877e-02,
-4.9647e-01, -5.1489e-02, -3.8899e-02,  2.5704e-01, -1.3399e-01,
 6.3114e-01, -1.4173e-01,  5.6961e-01,  1.0953e-01, -1.0889e-02,
-3.9217e-01, -2.7488e-01,  3.7711e-02, -1.1228e-01, -4.0066e-01,
 2.0202e-01, -5.6528e-02,  4.8125e-01,  2.3786e-01, -2.5096e-01,
-4.7954e-01,  7.9138e-01, -1.4246e-01,  2.8228e-01, -2.1925e-01,
 5.3708e-01,  2.9595e-01, -3.8215e-01,  2.0211e-01,  4.4877e-02,
-1.7880e-01,  1.3182e-01, -7.3205e-02, -2.4110e-01, -5.1567e-01,
-3.8671e-01,  2.9665e-01, -7.5373e-02,  2.8324e-01, -4.9246e-01,
 2.5574e-01,  1.8671e-01,  5.0763e-01,  6.0603e-01, -3.9349e-01,
 3.7441e-01,  1.9055e-01,  2.5257e-02,  2.3391e-01,  2.5466e-02,
 1.2332e-01, -7.4791e-02, -1.1984e-01, -3.4361e+00,  3.8000e-01,
 4.6583e-01, -3.9300e-01,  5.0151e-01, -2.4369e-01, -2.9067e-01,
-2.1698e-01, -7.6502e-01,  4.3197e-01,  3.6255e-02, -5.0161e-01,
 4.6126e-01,  3.4883e-02,  5.8544e-01, -5.8060e-01, -8.0063e-02,
-2.2829e-01, -5.8635e-01,  2.6740e-01, -2.4902e-01, -2.9530e-01,
 1.9320e-01, -4.0031e-01,  3.4973e-01,  3.1015e-01,  1.7396e-01,
-6.2707e-01, -6.4691e-02, -1.6592e-01, -2.2870e-02, -3.4658e-01,
 2.8630e-01,  2.9902e-01, -6.8322e-02,  3.2539e-01,  2.1884e-01,
-4.4088e-02, -5.9315e-02,  3.3574e-01,  2.5083e-01, -1.3336e-01,
 1.3482e-01,  1.9703e-01,  3.6615e-01,  8.1637e-03,  1.3348e-02,
-6.1421e-01,  4.2838e-01,  2.5890e-01,  7.8725e-02,  1.6269e-01,
-5.2337e-01,  3.1335e-02, -1.4687e-01,  5.3146e-02,  1.3526e-01,
 2.6760e-01, -4.2279e-01, -2.9381e-01,  2.6565e-01, -2.1339e-01,
-2.6541e-01,  4.0703e-01,  7.2949e-01, -3.1052e-01, -4.0835e-01,
-3.0879e-01, -1.9953e-01,  2.2560e-01, -2.0542e-02, -1.1085e-01,
-4.4369e-02, -1.0910e+00,  1.4962e-01, -3.0682e-01,  4.5071e-01,
-2.3607e-01, -1.5306e-01, -3.3796e-01,  6.5300e-02, -3.6088e-01,
 1.1439e-01, -1.2330e-01,  3.4044e-02, -8.3588e-01, -1.3427e-01,
-1.8076e-02, -1.3784e-01, -7.4650e-01,  2.0982e-01,  4.2198e-01,
 2.7729e-02,  2.9784e-01, -1.2599e-02, -2.3629e-02,  1.8879e-01,
-2.6840e-01,  1.8048e-02, -3.2483e-01,  2.9658e-01, -7.4130e-02,
 1.8513e-01,  2.3700e-01,  1.7753e-01,  2.1279e-01, -6.0714e-01,
 1.1632e-01,  1.7223e-01,  5.1921e-02, -1.3206e-01, -8.2450e-01,
```

```
 4.7174e-01, -4.7895e-01, -1.1375e-01, -5.9135e-03,  7.7163e-01,
 8.8599e-01, -7.9604e-02,  2.5579e-01, -1.8789e-01,  4.7801e-01,
-2.7655e-01, -3.7498e-01,  2.4066e-01,  1.2196e-01, -1.4290e-01,
-2.9800e-01, -8.2154e-02, -4.5267e-01,  1.9302e-03, -7.0471e-02,
 3.0694e-02, -7.2871e-02, -5.1416e-02,  3.8583e-01, -2.6367e-01,
-4.5985e-01,  1.6101e-01,  1.8223e-01, -2.8691e-01,  4.8425e-01,
 3.3665e-01, -2.2698e-01,  3.3842e-01,  8.3098e-02,  2.3594e-01,
 2.8146e-01,  6.8816e-02,  2.9626e-01, -3.4740e-01,  4.8083e-02,
 2.4245e-02, -3.1148e-01, -1.2181e-01, -4.0197e-01,  4.8986e-01,
-6.2500e-02,  7.9493e-02, -3.0729e-01, -2.3482e-01, -3.1585e-02,
 2.6604e-01, -1.2823e-02, -1.6638e-01,  5.9506e-01,  6.5870e-01,
-2.6763e-01, -1.5464e-01,  3.2974e-01, -1.9271e-01, -4.5264e-01,
-1.4018e-02,  3.7222e-01, -5.4768e-01,  4.8440e-01, -2.7273e-01,
-3.7305e-01, -4.6690e-01,  4.0682e-01,  1.6783e-02, -3.1254e-01,
-1.2740e-01, -1.5137e-01,  4.7583e-01, -2.6754e-02,  1.7162e-01,
-8.9333e-02,  4.3226e-01, -3.7488e-02, -1.1128e-01,  7.1107e-01,
-2.8997e-02,  1.3905e-01, -7.4379e-01, -1.1332e-01, -6.2653e-02,
 1.8076e-01,  3.4005e-01,  3.1012e-02,  5.9343e-02,  2.2329e-02,
 1.3199e-01, -2.0150e-01, -1.8737e-01, -2.5710e-01, -3.8929e-02,
-1.9592e-01, -1.3384e-01, -7.8204e-02, -3.7188e-01, -4.3669e-01,
-1.0618e-02, -2.8074e-01,  1.1675e-01,  4.1888e-01, -1.6704e-02,
-3.3174e-01, -2.6033e-01,  3.4215e-01, -2.9482e-01, -2.3508e-01,
 1.5812e-02, -4.5972e-04, -4.5515e-01, -2.6120e-01, -5.2407e-01,
 3.6386e-01, -1.0274e-02, -2.1431e-01,  4.1656e-01, -1.2866e-01,
-3.8174e-01, -2.8323e-01, -6.7305e-02,  2.3405e-01, -1.7518e-01,
-8.4004e-01,  2.8643e-01,  3.3858e-01,  4.6143e-01,  5.0825e-01,
 3.4562e-01,  3.2663e-01,  3.8611e-01,  2.2974e-01, -1.8653e-01,
 3.0548e-01, -2.5984e-01, -2.1290e-01,  2.7246e-01,  6.1730e-01,
 1.4782e-01,  1.3325e-01, -1.2792e-01, -5.5985e-02, -6.9295e-01,
-1.0010e-01,  1.5259e-01,  2.5327e-02,  3.6308e-02, -3.0183e-01,
-2.1349e-01,  4.9935e-01,  2.4388e-01, -1.6569e-01, -2.2274e-01,
 6.1269e-01,  3.7776e-01,  3.0784e-01, -1.3477e-01, -1.8492e-01,
 3.5202e-01, -1.0704e-01, -1.3111e-01,  1.8026e-01, -1.1725e-01,
-6.4544e-01, -4.4672e-01, -1.3389e-01, -1.8063e-01,  5.2438e-01,
-3.9345e-01,  8.5842e-02, -2.3086e-02, -2.3888e-01,  3.3303e-02,
 3.1724e-01, -3.2123e-01,  1.7069e-01,  3.1929e-01,  4.5155e-01,
 1.4722e-01, -7.4359e-02,  3.1483e-02,  6.2563e-02,  7.6846e-01,
 4.8866e-01, -1.9362e-01,  2.5538e-01, -7.4458e-01,  2.8427e-01,
 2.8396e-01, -3.4864e-01,  6.6814e-02, -3.4439e-01, -2.0368e-01,
-1.1654e-02, -5.2065e-01,  3.8175e-01, -3.4459e-01, -1.7131e-01,
 2.6277e-01,  6.2026e-01, -3.9878e-01,  4.0546e-01, -4.4139e-01,
-4.2970e-01, -1.0945e-03,  4.0683e-01,  6.9765e-02,  6.7103e-03,
 3.7868e-01,  1.1418e-01,  8.1064e-02,  2.5002e-01, -3.2178e-01,
 4.0234e-01, -1.3525e-01,  4.4883e-02, -1.4421e-01,  1.1563e-01,
 9.4357e-02,  4.6874e-01,  9.7760e-02, -2.1971e-01,  5.6885e-01,
 4.8612e-01,  8.3830e-03, -2.7764e-01,  3.1705e-01, -8.6870e-02,
-1.8225e-01,  2.8047e-01,  4.9300e-02, -3.8422e-01,  3.5691e-01,
 1.7433e-01,  3.6039e-01,  1.1028e-01,  2.5789e-01,  4.7826e-01,
 2.3553e-01,  1.0329e-02,  6.4360e-02, -9.8696e-02,  3.0023e-01,
 1.5701e-01, -4.8440e-02,  2.6268e-01,  1.2060e-01,  7.3864e-01,
 2.0730e-01, -4.5226e-01,  1.1489e-01,  2.7837e-01,  1.0547e-01,
-6.2836e-01, -1.4813e-01, -6.1692e-02,  4.6081e-01,  6.1109e-01,
-7.5599e-02, -1.5136e-01,  1.6815e-01,  3.1557e-01, -2.9714e-01,
-2.5764e-01, -5.3547e-01,  2.9376e-01, -5.3005e-01, -1.6957e-02,
-2.9038e-01, -5.2431e-01, -9.9385e-02,  6.5450e-02, -2.2260e-01,
-7.8313e-01, -1.0781e-01, -3.7012e-01, -2.6512e-01, -2.5467e-01,
 4.0734e-01, -4.1198e-01,  8.7190e-02, -4.6659e-01,  6.8911e-01,
 4.8816e-01,  3.7634e-01,  3.0287e-01,  6.2624e-01, -5.7808e-01,
-4.0052e-01,  2.3407e-01,  4.3606e-01, -4.7639e-01,  1.1819e-02,
-1.7437e-02, -3.1608e-01,  3.7167e-02,  5.5680e-01,  2.6963e-01,
```

```
          -4.7000e-01,  3.9094e-01, -1.4054e-01,  3.2537e-03,  1.4709e-01,
          -2.8214e-01, -5.2028e-01, -2.7766e-01,  1.6080e-01, -2.3658e-01,
           2.7423e-01, -3.8823e-02, -2.8725e-01,  6.7195e-02,  3.4809e-01,
          -4.4231e-02, -9.5801e-03,  1.3890e-01, -3.0075e-01,  1.2015e-01,
           1.6740e-01, -2.8660e-01,  1.9426e-02,  6.5305e-02, -1.1547e-01,
           5.0343e-02, -5.2206e-01, -6.3841e-01,  4.7499e-01,  1.2367e-01,
           3.1301e-01, -9.4102e-02,  4.9416e-02, -1.0102e-01, -2.5378e-02,
           1.1833e-01, -1.8191e-01, -6.0487e-01, -1.5069e-01, -5.3343e-02,
          -5.6242e-01, -1.2877e-02, -1.5974e-02,  1.8602e-01,  1.2440e-01,
          -1.7414e-01,  1.7107e-01,  2.2009e-01]])
Embedding for sentence 3: tensor([[-1.1238e-01, -6.2890e-02, -4.6874e-01, -1.3339
e-01, -1.5828e-01,
           3.1799e-01,  7.2418e-04,  7.5508e-01, -1.9639e-01, -2.4970e-01,
          -1.6502e-01, -2.6042e-01, -1.8176e-01,  3.1048e-01, -3.6831e-01,
           3.6051e-01,  2.4238e-01,  4.3221e-01, -1.7785e-01,  5.1978e-01,
           9.2652e-02,  1.0910e-02, -6.1341e-01, -6.9609e-03,  7.2848e-01,
          -1.9996e-01, -1.2920e-01,  5.3205e-02,  1.4341e-01, -1.1997e-01,
           2.2208e-01,  1.3069e-01,  4.2986e-01,  2.4119e-01,  1.2204e-02,
          -4.7576e-01,  9.7546e-02, -1.9431e-01, -2.9555e-01,  6.1339e-02,
          -2.2666e-01, -3.3094e-01,  1.8950e-01, -1.4407e-01,  1.0781e-01,
          -2.6928e-01, -1.4800e-01,  2.8953e-01, -1.8820e-01,  2.1850e-01,
          -5.2174e-01,  1.1353e-01, -6.5858e-02,  1.8141e-01,  1.5890e-01,
           5.0042e-01,  1.1311e-01,  5.7517e-02,  1.7110e-01, -4.1462e-01,
          -1.2530e-01, -1.0491e-01,  2.5830e-02, -8.1439e-02,  2.5352e-02,
           2.2271e-01, -8.1642e-02,  2.9166e-01, -7.7009e-01,  1.1622e-01,
          -3.4971e-01, -2.5309e-01,  1.0649e-01, -4.2235e-01,  6.9649e-03,
          -2.2571e-01, -3.0149e-01, -1.3146e-01, -7.1883e-02, -1.4306e-01,
          -4.7248e-01,  2.2011e-01, -2.3985e-02,  1.8323e-01, -3.2564e-03,
           8.6137e-02, -3.8505e-01, -2.2014e-01, -2.4963e-01,  5.3523e-01,
          -3.6447e-01, -4.0790e-02,  1.1901e-01,  5.7837e-01, -1.6721e-02,
          -1.9843e-01,  8.6654e-02,  3.7128e-02,  4.1931e-01,  1.6164e-01,
           5.2744e-02, -4.3392e-01,  2.7536e-01,  5.0015e-01, -4.0545e-01,
          -1.6745e-01, -1.8306e-01,  1.1774e-01,  2.5691e-01, -3.6054e-01,
           3.7216e-01, -5.1270e-01,  3.0475e-01,  3.9664e-01, -4.5763e-01,
           5.4955e-01, -2.2760e-02,  1.8245e-02,  1.8081e-02, -1.7399e-01,
          -3.2932e-01, -1.7879e-01,  3.3060e-01,  4.0007e-01, -4.6469e-02,
          -1.1570e-01, -1.5458e-01,  1.6501e-01, -3.9006e-02, -3.4964e-01,
           8.5408e-02,  1.7973e-01, -7.5952e-02, -9.3711e-03, -2.7851e-01,
           5.5231e-02,  2.1780e-01, -6.2260e-02, -1.8031e-01, -2.3030e-01,
          -3.0393e-01,  5.8980e-01,  2.8375e-01, -2.7968e-01,  3.6204e-01,
          -1.1395e-01, -1.0975e-01, -1.8106e-02,  3.7155e-03, -1.0500e-02,
           3.3397e-01,  3.3594e-01, -2.8235e-01,  1.0719e-02, -2.4796e-01,
           2.4623e-01, -3.8655e-01,  2.0793e-01,  5.8677e-02,  1.3425e-01,
           1.4840e-01, -1.8332e-01,  1.2225e-01, -1.9570e-02, -3.1717e-01,
           2.8363e-01, -3.4722e-01,  3.5650e-01, -2.9572e-01,  4.5957e-01,
          -7.4128e-02, -9.8055e-02,  9.7943e-01,  2.0624e-01, -2.6378e-01,
           3.2810e-01, -2.5085e-01,  6.5790e-02,  2.9116e-01,  5.3333e-02,
          -1.5401e+00,  5.3473e-01,  6.5843e-01,  2.1874e-01, -3.6980e-01,
          -1.8384e-01, -3.4166e-01, -5.1099e-01,  2.6764e-01,  3.7277e-01,
          -1.9712e-01, -4.0630e-01, -6.3927e-01, -1.4702e-01, -2.7961e-01,
          -6.9512e-01, -1.3468e-02, -5.1627e-01, -2.6445e-01, -9.2562e-02,
           3.8772e-02,  5.5500e-02,  2.0379e-01,  2.6627e-01,  6.7232e-02,
           6.2958e-01,  3.0026e-01, -2.1268e-01, -6.2376e-02,  3.5979e-02,
          -3.5941e-01,  3.0550e-01,  4.2949e-01, -2.9920e-01,  4.4585e-01,
           1.5071e-01,  8.8059e-02, -1.2497e-01, -3.9817e-01, -1.8652e-01,
           1.1595e-01, -1.6918e-01, -5.5742e-01,  2.8582e-01,  8.2971e-03,
           2.6931e-01,  3.3328e-01,  3.4692e-01,  9.9388e-02,  1.0542e-01,
           2.7126e-01, -3.2948e-01,  4.0767e-01,  2.0448e-01, -4.6623e-01,
           4.2273e-02, -8.3718e-02, -1.7216e-01,  3.1239e-01, -2.4973e-01,
          -3.0876e-01,  1.8742e-01,  3.0809e-01, -1.0199e-01, -6.9005e-02,
```

```
-3.8685e-01,  9.8772e-03, -1.2391e-01, -4.8620e-02, -2.0661e-01,
 9.8676e-03, -5.5816e-01,  1.6176e-01, -3.6870e-02, -1.2452e-01,
-8.1385e-02,  1.9142e-01,  3.2465e-01,  3.3195e-01, -2.2435e-02,
-4.0097e-02,  2.0324e-01,  5.5819e-01, -3.8881e-02, -2.5674e-01,
 3.4286e-01, -8.4377e-02,  4.7665e-01,  2.0887e-01,  1.5017e-02,
-9.6931e-02, -1.2315e-01, -1.9171e-01, -1.8283e-01, -7.2681e-01,
-3.9840e-01,  4.0322e-01, -8.9597e-04, -2.2116e-01,  1.6081e-01,
 5.7423e-02,  3.8623e-01, -4.3406e-01,  1.8944e-01,  3.9003e-01,
-4.1448e-01,  2.6065e-01, -3.2635e-01, -2.1611e-01, -3.1958e-01,
-4.1957e-01,  7.0687e-01,  1.6784e-02, -2.7031e-01,  4.9940e-01,
 1.3181e-01, -4.6448e-02,  5.0741e-01,  1.0025e-01, -3.2606e-01,
 4.0131e-01, -2.9066e-01,  4.6084e-02, -5.6380e-02, -2.0619e-01,
 4.2690e-01, -1.4050e-01, -2.3188e-01, -3.6589e+00,  9.4184e-02,
-3.0229e-01, -4.5971e-01,  9.2848e-02, -2.0841e-01, -9.6476e-02,
-4.1314e-01,  2.5739e-02, -2.5964e-01, -7.7343e-02, -5.6753e-01,
 2.0834e-01, -1.7319e-02,  1.7707e-01, -5.2924e-02,  7.9023e-02,
-5.1502e-01, -2.2822e-01,  3.9121e-01, -2.3649e-01,  1.0081e-01,
 2.9077e-01, -1.4671e-01,  1.1053e-02,  3.0127e-01, -1.0423e-01,
-1.5066e-01, -3.5045e-01, -2.8069e-01, -1.4040e-01, -1.4420e-01,
 6.6752e-02, -4.3915e-02,  2.9964e-01, -9.0526e-02, -2.5158e-01,
-3.1626e-01,  1.8282e-01,  2.9300e-01, -1.2016e-01,  4.2027e-01,
 2.0578e-01,  1.7996e-01,  6.5164e-01, -8.4013e-02,  3.6891e-01,
-3.9884e-01,  1.2438e-01,  4.1233e-01,  1.3214e-01,  2.2817e-01,
 3.3701e-02, -3.5518e-02,  1.5799e-01,  4.7337e-01,  3.6536e-01,
 1.4593e-01, -1.4050e-01, -6.7957e-01,  6.7384e-02, -2.5669e-01,
-1.3770e-01,  3.2208e-01,  5.5413e-03, -3.4829e-01, -2.9045e-01,
-1.4710e-02,  8.8730e-02,  1.0725e-01, -2.0145e-01, -1.9012e-01,
-2.1724e-01, -1.1634e+00, -3.1143e-01, -4.1638e-01, -5.4134e-02,
 2.0427e-03,  4.0366e-01,  3.5698e-02, -4.3734e-01, -2.8341e-01,
-5.0976e-02,  8.5088e-02, -1.9635e-01, -2.8477e-01,  1.4562e-01,
 8.3084e-02, -3.6482e-01, -7.7753e-02,  2.3591e-02,  1.7935e-01,
 1.2867e-01, -9.9530e-02, -8.2701e-02, -3.9396e-03,  3.2572e-01,
-6.7959e-01,  1.5187e-03, -3.2342e-01, -2.7843e-01, -3.0307e-01,
 2.6645e-01,  1.5458e-01,  3.1230e-01,  3.6094e-01, -3.1598e-01,
-1.0138e-01,  2.3798e-01,  2.3291e-01,  6.7818e-01, -4.8848e-01,
 5.0534e-01, -1.4881e-01, -1.1483e-02,  2.9094e-02,  7.3971e-02,
 3.3548e-01,  2.8412e-01,  2.5172e-01, -3.0995e-02,  5.2830e-01,
-2.9101e-01, -4.7133e-01, -6.0311e-01,  3.3996e-02, -9.6570e-02,
-6.0649e-01, -2.3960e-01,  2.9497e-01, -3.0266e-01, -1.6308e-01,
-1.2283e-01,  1.9447e-01, -3.8723e-03, -1.3976e-03, -6.9212e-01,
-2.4113e-02, -8.5837e-02,  2.9075e-01,  1.6620e-01,  2.4529e-01,
-2.5292e-01, -5.0481e-02,  3.3450e-02,  3.5677e-01,  9.1811e-02,
-9.9125e-02, -4.9066e-02,  2.7896e-01,  1.5011e-02, -1.1892e-01,
-1.7880e-01, -5.5551e-01, -3.9367e-01, -6.0220e-01,  2.8321e-01,
-3.4488e-01, -4.0714e-01, -4.0461e-01, -3.5709e-02, -1.0679e-01,
 3.3417e-01, -1.4069e-01, -7.2565e-03,  6.6518e-01,  3.5813e-01,
 2.0531e-01, -3.2911e-01,  5.8608e-01, -1.1795e-01, -1.9804e-01,
 1.3518e-01,  5.0807e-01, -1.7510e-01,  3.5008e-01,  3.1587e-01,
-3.8473e-01, -9.8679e-02,  2.7254e-01, -6.7117e-02, -3.7770e-01,
-1.1323e-01,  1.2740e-01,  5.4334e-02,  1.5096e-01, -6.1051e-02,
-1.0747e-01,  2.1980e-01,  1.0671e-01,  2.2827e-03,  1.4665e-01,
-4.2908e-01, -2.9436e-01, -6.5440e-01,  6.5411e-02, -1.2737e-01,
-2.2501e-01,  5.9202e-01,  1.9123e-01,  1.4794e-01, -2.4933e-01,
 4.6599e-02,  3.2406e-01, -3.2464e-01,  2.5002e-01,  4.5560e-01,
 1.2563e-01, -6.7571e-01, -1.4601e-01, -3.2224e-01, -4.4461e-01,
-7.1287e-02, -6.2670e-01, -5.3398e-02,  1.2434e-01, -1.3088e-01,
 5.5635e-02,  9.8389e-03, -3.1196e-01, -3.3982e-01, -2.8896e-01,
 5.3853e-02, -1.1904e-01,  1.1611e-01, -6.0543e-03, -1.3530e-01,
-1.9179e-01, -1.7998e-01,  3.1532e-01,  5.7094e-01,  2.8510e-02,
 1.0379e-02,  5.3015e-02,  3.2561e-01,  1.4631e-01, -5.5848e-01,
```

```
-4.8763e-01,  2.8659e-01,  8.8966e-02,  4.5759e-01, -3.0510e-01,
-1.6016e-01, -9.9715e-02, -2.1058e-01, -1.7291e-01,  3.6265e-02,
 4.6210e-01, -2.5199e-01,  5.0573e-01, -2.2774e-01,  2.6486e-01,
-3.1831e-02, -5.4563e-01, -1.4822e-01, -2.4404e-01, -4.5950e-01,
 1.9129e-01,  4.1922e-02,  2.4493e-02,  1.6393e-01,  8.8925e-02,
 3.1395e-01,  4.7534e-01, -2.9106e-02,  1.5713e-01,  8.2795e-02,
 1.7500e-01,  3.5900e-01,  2.8068e-01,  2.4732e-01, -4.0425e-02,
 4.4615e-01,  1.6198e-01, -5.7148e-02, -3.2332e-02, -2.9100e-01,
-7.2083e-01, -8.2075e-02, -2.4817e-01,  1.7838e-01,  5.7392e-01,
-2.3874e-02,  1.7843e-01,  4.1431e-01, -1.4548e-01, -7.6240e-02,
 5.2127e-01, -4.3383e-01, -7.7471e-02,  3.6934e-01,  1.8708e-01,
 6.0578e-02,  2.3650e-01, -1.0388e-02,  8.3833e-02,  5.6368e-02,
 1.7575e-01, -3.8310e-01, -1.1383e-01, -3.5840e-01,  7.8178e-01,
 5.1333e-01,  1.0382e-02, -2.9741e-01,  3.4013e-01, -1.4739e-01,
-5.7701e-01,  2.1030e-01,  2.8422e-01, -3.1136e-01, -1.5186e-01,
 2.6909e-01,  5.8814e-01, -8.3292e-02, -3.7748e-02,  2.5405e-01,
-3.3427e-01, -1.4573e-01,  5.4811e-01,  3.9009e-01, -3.7674e-01,
 3.1882e-01, -1.2552e-01, -7.8280e-02,  3.2873e-01, -3.3190e-01,
 1.6305e-02, -2.4208e-01,  3.2553e-01, -1.7931e-01,  6.0535e-01,
-2.7503e-01,  3.4586e-01, -8.9133e-03, -3.8137e-02, -2.5696e-02,
 1.4994e-01, -3.5965e-03,  1.7049e-01,  5.2830e-01,  3.8703e-01,
 1.5012e-01,  8.2804e-02,  3.9701e-01,  8.3273e-02, -3.3035e-02,
-2.0067e-01,  4.2045e-01,  3.7521e-01,  4.7172e-01,  1.7525e-01,
 1.3215e-01, -2.0608e-01,  9.5133e-02,  1.2269e-01,  1.4750e-01,
 1.8825e-01, -2.4750e-01,  1.5914e-03, -1.2374e-01,  4.5559e-01,
 6.0098e-02, -2.8746e-01,  1.4040e-02,  5.5030e-01, -2.7882e-01,
-3.0215e-01, -6.0805e-01,  3.8725e-01,  8.4131e-02,  9.7369e-02,
 1.0292e-01,  3.9680e-01, -3.1952e-02,  4.0168e-01, -5.6698e-01,
 3.1457e-01, -2.2693e-01,  1.6128e-01, -1.7119e-01, -2.7648e-02,
-5.7962e-01, -3.9685e-01, -2.9200e-01, -1.3669e-01, -5.5789e-01,
-2.8466e-01, -1.4642e-01,  1.6908e-02, -4.9734e-02,  2.1155e-01,
 1.9851e-01, -4.6078e-01, -3.7184e-02, -1.4421e-01,  1.3807e-01,
 5.9214e-01,  2.0069e-01, -9.9616e-02,  2.2810e-01, -8.5267e-02,
-8.1441e-02,  4.5350e-02, -2.9271e-01, -9.7491e-02,  3.1083e-01,
 4.6460e-01, -3.5562e-01, -4.1695e-01,  3.8212e-01,  3.9255e-01,
-8.7739e-01, -6.1844e-03,  2.2264e-01,  1.3129e-02,  1.2494e-02,
 2.3933e-01, -1.2377e-01, -1.9634e-01, -4.2990e-02, -3.9516e-02,
 1.4875e-01,  6.5889e-01, -4.9134e-01, -6.7860e-02,  2.0189e-01,
 7.5436e-01, -3.9327e-01, -3.5464e-01, -2.2844e-01,  6.2945e-01,
 7.2747e-02, -2.8051e-01, -2.0827e-01, -2.8315e-01,  1.1601e-01,
-9.8426e-02, -5.6247e-01, -5.9988e-02, -6.4766e-02, -2.0143e-01,
-1.2348e-01, -1.3073e-01, -1.1713e+00,  8.4172e-02,  1.4402e-01,
 2.0905e-01, -8.4438e-02, -3.7452e-01, -9.9875e-02, -1.6433e-01,
 2.9854e-01,  1.9293e-01,  2.0256e-01, -1.6964e-01, -6.3707e-02,
 1.2543e-01,  2.2712e-01,  7.4110e-03]])
```

In [ ]: