Sittichai Chaikamol

Word Embeddings

Introduction

This reflection journal focuses on the experience with GloVe Word Vectors, a powerful tool for understanding language. It represents words as numbers in a high-dimensional space.

The goal of this reflection is to explore personal involvement and learning with GloVe Word Vectors, highlighting what was learned and difficulties faced during the lab activity.

Background Information

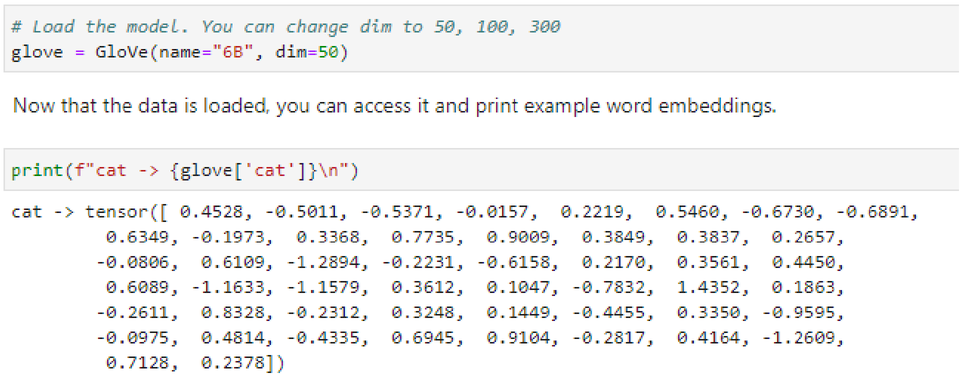
Word embeddings like Word2Vec, FastText, and GloVe help computers understand words by turning them into lists of numbers. This makes it easier for machines to see how words are related.

GloVe Word Vectors are a way to turn words into numbers, making it easier for analysis to understand language. Each word gets a vector, a list of numbers that show its meaning. Words with similar meanings have vectors that are close to each other. This happens by analyzing lots of text to see which words appear together often, helping to figure out how words are related.

The dataset that was used in the lab was already pre-trained by someone else. This saves a lot of time and effort. In the lab, these vectors were looked at to see how words relate to each other. The closeness of the vectors of different words was measured using something called cosine similarity, which shows if words are similar.

Specific Details:

The process began by loading a pre-trained GloVe model. This model represents words as vectors with 50 dimensions, simplifying the handling and comprehension of large datasets without requiring us to train the model ourselves. For example, we can quickly get the vector for the word "cat" like this:



```
# Load the model. You can change dim to 50, 100, 300
glove = GloVe(name="6B", dim=50)
```

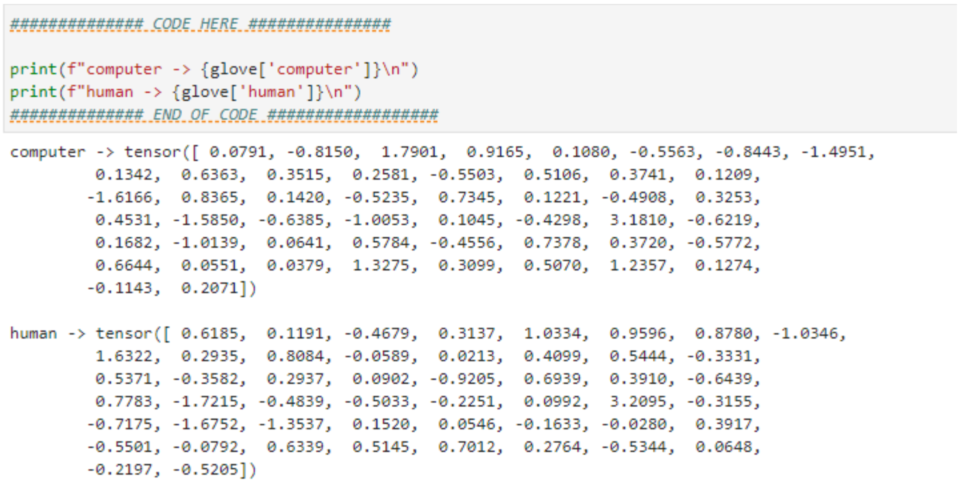
Now that the data is loaded, you can access it and print example word embeddings.

```
print(f"cat -> {glove['cat']}\n")
```

```
cat -> tensor([ 0.4528, -0.5011, -0.5371, -0.0157,  0.2219,  0.5460, -0.6730, -0.6891,
         0.6349, -0.1973,  0.3368,  0.7735,  0.9009,  0.3849,  0.3837,  0.2657,
        -0.0806,  0.6109, -1.2894, -0.2231, -0.6158,  0.2170,  0.3561,  0.4450,
         0.6089, -1.1633, -1.1579,  0.3612,  0.1047, -0.7832,  1.4352,  0.1863,
        -0.2611,  0.8328, -0.2312,  0.3248,  0.1449, -0.4455,  0.3350, -0.9595,
        -0.0975,  0.4814, -0.4335,  0.6945,  0.9104, -0.2817,  0.4164, -1.2609,
         0.7128,  0.2378])
```

This code shows us the 50-dimensional vector for "cat," which is a list of numbers that represents its meaning in a way that computers can process.

Next, involved tackling challenge where we generated vectors for "computer" and "human" using the same pre-trained GloVe model. The code I wrote for this part was:

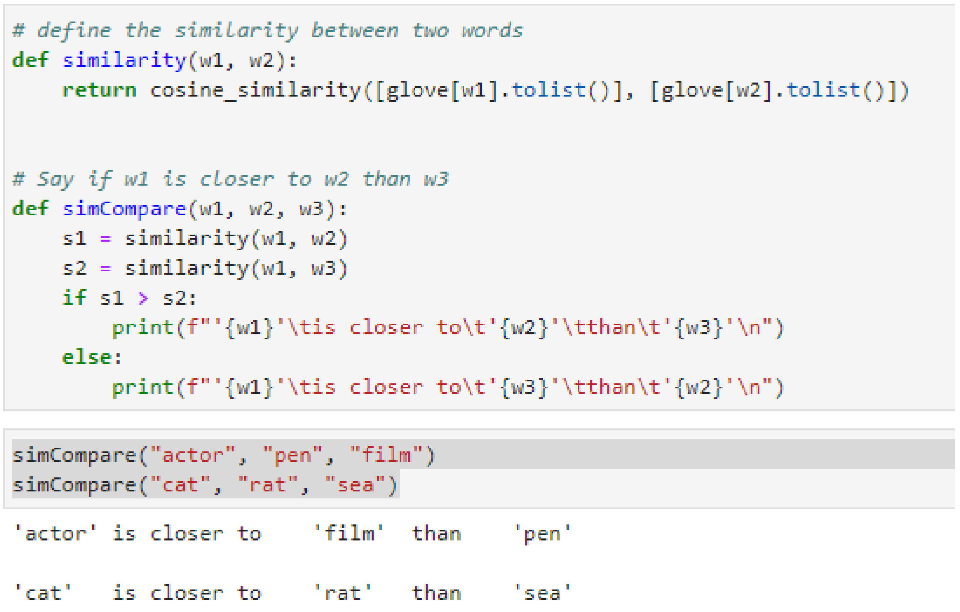


```
############## CODE HERE ###############

print(f"computer -> {glove['computer']}\n")
print(f"human -> {glove['human']}\n")
############## END OF CODE ##################
```

```
computer -> tensor([ 0.0791, -0.8150,  1.7901,  0.9165,  0.1080, -0.5563, -0.8443, -1.4951,
         0.1342,  0.6363,  0.3515,  0.2581, -0.5503,  0.5106,  0.3741,  0.1209,
        -1.6166,  0.8365,  0.1420, -0.5235,  0.7345,  0.1221, -0.4908,  0.3253,
         0.4531, -1.5850, -0.6385, -1.0053,  0.1045, -0.4298,  3.1810, -0.6219,
         0.1682, -1.0139,  0.0641,  0.5784, -0.4556,  0.7378,  0.3720, -0.5772,
         0.6644,  0.0551,  0.0379,  1.3275,  0.3099,  0.5070,  1.2357,  0.1274,
        -0.1143,  0.2071])

human -> tensor([ 0.6185,  0.1191, -0.4679,  0.3137,  1.0334,  0.9596,  0.8780, -1.0346,
         1.6322,  0.2935,  0.8084, -0.0589,  0.0213,  0.4099,  0.5444, -0.3331,
         0.5371, -0.3582,  0.2937,  0.0902, -0.9205,  0.6939,  0.3910, -0.6439,
         0.7783, -1.7215, -0.4839, -0.5033, -0.2251,  0.0992,  3.2095, -0.3155,
        -0.7175, -1.6752, -1.3537,  0.1520,  0.0546, -0.1633, -0.0280,  0.3917,
        -0.5501, -0.0792,  0.6339,  0.5145,  0.7012,  0.2764, -0.5344,  0.0648,
        -0.2197, -0.5205])
```

This gave us the vectors for "computer" and "human," helping us see how the model represents these words numerically.

Then, a technique called Cosine Similarity, a method to measure how similar two-word vectors are. This method calculates the cosine of the angle between two vectors, giving us a score that tells us how closely related two words are. The code below was used to compare words:



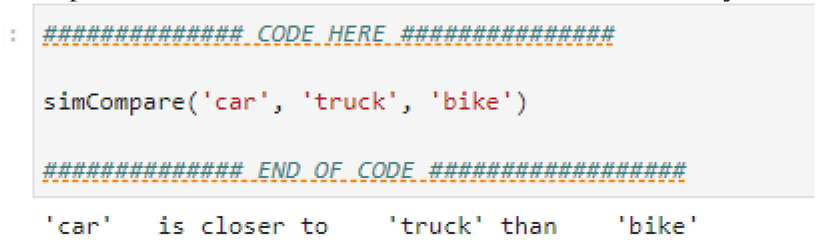
```
# define the similarity between two words
def similarity(w1, w2):
    return cosine_similarity([glove[w1].tolist()], [glove[w2].tolist()])


# Say if w1 is closer to w2 than w3
def simCompare(w1, w2, w3):
    s1 = similarity(w1, w2)
    s2 = similarity(w1, w3)
    if s1 > s2:
        print(f"'{w1}'\tis closer to\t'{w2}'\tthan\t'{w3}'\n")
    else:
        print(f"'{w1}'\tis closer to\t'{w3}'\tthan\t'{w2}'\n")
```

```
simCompare("actor", "pen", "film")
simCompare("cat", "rat", "sea")
```

```
'actor' is closer to    'film'  than    'pen'

'cat'   is closer to    'rat'   than    'sea'
```

This function lets us see which of two words is more similar to a third word, helping us understand the relationships between words based on their vectors.

The last challenge was to determine if "car" is closer to "truck" than to "bike." This was a practical about GloVe vectors and Cosine Similarity.



```
############### CODE HERE ################

simCompare('car', 'truck', 'bike')

############### END OF CODE ###################
```

```
'car'   is closer to    'truck' than    'bike'
```

Personal Reflection

Thoughts and Feelings:

- Turning words into numbers seemed odd at first. But as I followed the steps in the lab, it became clearer and quite fascinating.

Analysis and Interpretation:

- Working with the GloVe model was a hands-on way to see how word relationships are mapped out by computers. It was exciting to see this in action and made the topic more tangible for me.

Connections to Theoretical Knowledge:

- The lab turned the concepts we've learned into practical experiences. It demonstrated the significance of word embeddings and their application in real tasks.

Critical Thinking:

- The lab also made me think about the challenges and limitations of using pre-trained models like GloVe. It raised questions about how these models are trained and their accuracy.

Discussion of Improvements and Learning

Personal Growth:

- This lab significantly enhanced my grasp of word embeddings, providing me with valuable insights into their application and helping me understand their role in natural language processing (NLP).

Skills Developed:

- I learned how to use the GloVe model to explore word vectors and gained practice in calculating the similarity between words with cosine similarity. These are valuable skills for any NLP project.

Future Application:

- The experience from this lab will be useful for future projects, especially those involving text analysis. I look forward to applying these new skills in other contexts.

Conclusion

Reflecting on my lab with GloVe Word Vectors, I've gained new insights into word embeddings and their practical application in NLP. The lab was a key step in linking theory with practice. Exploring GloVe embeddings has not only improved my technical skills but also sparked further interest in NLP. I'm eager to explore more applications of these techniques.

Sittichai Chaikamol

References:

- https://www.youtube.com/watch?v=5MaWmXwxFNQ
- https://www.youtube.com/watch?v=s6qK2-ypagE
- AWS Module 2 Lab 3