Section 1: Properties of Word Embeddings

**Question 1.1** Read the documentation of the Vocab class of Torchtext that you can find here: https:// torchtext.readthedocs.io/en/latest/vocab.html and then read the A1\_Section1\_starter.ipynb code. Run the notebook and make sure you understand what each step does.

**Question 1.2** Write a new function, similar to print\_closest\_words called print\_closest\_cosine\_words that prints out the N-most (where N is a parameter) similar words using cosine similarity rather than euclidean distance. Provide a table that compares the 5-most cosine-similar words to the word ‘dog’, in order, alongside to the 10 closest words computed using euclidean distance. Give the same kind of table for the word ‘computer.’ Looking at the two lists, does one of the metrics (cosine similarity or euclidean distance) seem to be better than the other? Explain your answer. Submit the specific code for the print\_closest\_cosine\_words function that you wrote in a separate Python file named A1P1\_2.py. [2 points]

|  |  |  |  |
| --- | --- | --- | --- |
| **Word** | **Cosine Similarity** | **Word** | **Euclidean Distance** |
| Cat | 0.92 | Cat | 1.88 |
| Dogs | 0.85 | Dogs | 2.65 |
| Horse | 0.79 | Puppy | 3.15 |
| Puppy | 0.78 | Rabbit | 3.18 |
| pet | 0.77 | Pet | 3.23 |
|  |  | Horse | 3.25 |
|  |  | Pig | 3.39 |
|  |  | Pack | 3.43 |
|  |  | Cats | 3.44 |
|  |  | Bite | 3.46 |

Table 1.2.1: similar words to “dog” by embedding.

|  |  |  |  |
| --- | --- | --- | --- |
| **Word** | **Cosine Similarity** | **Word** | **Euclidean Distance** |
| Computer | 0.92 | Computers | 2.44 |
| Software | 0.88 | Software | 2.93 |
| Technology | 0.85 | Technology | 3.19 |
| Electronic | 0.81 | Electronic | 3.51 |
| Internet | 0.81 | Computing | 3.60 |
|  |  | Devices | 3.67 |
|  |  | Hardware | 3.68 |
|  |  | Internet | 3.69 |
|  |  | Applications | 3.69 |
|  |  | Digital | 3.70 |

Table 1.2.2: similar words to “computer” by embedding.

For "dog": the cosine similarity gives us close semantic relationships like *dogs*, *puppy*, and *pet*, which all relate directly to animals or pets. Euclidean distance seems to capture words that are close in terms of numerical proximity, but they aren't always semantically related. For example, words like *pack* and *bite* are included in the Euclidean distance list for "dog" but don't have as strong a semantic relationship as those in the cosine similarity list.

For "computer": Cosine similarity shows that words like *software*, *technology*, and *internet* are highly related in meaning, while Euclidean distance provides a broader range of related terms, including *devices* and *hardware*.

To conclude, cosine similarity seems to be a better measure in this case because it captures semantic relationships more effectively. Euclidean distance, while useful for proximity in the vector space, might not always capture meaning as well.

**Question 1.3** The section of A1\_Section1\_starter.ipynb that is labelled Analogies shows how relation- ships between pairs of words is captured in the learned word vectors. Consider, now, the word-pair relationships given in Figure 1 below, which comes from Table 1 of the Mikolov[2] paper. Choose one of these relationships, but not one of the ones already shown in the starter notebook, and report which one you chose. Write and run code that will generate the second word given the first word. Generate 10 more examples of that same relationship from 10 other words, and comment on the quality of the results. Submit the specific code that you wrote in a separate Python file, A1P1\_3.py. [4 points]

A screenshot of a computer

Description automatically generated

Example 1.3: Demonstration of embedding recognizing pluralization of verbs.

For the 10 examples generated, the results are consistent. However, not all words exhibited this stability; some examples were unclear or nonsensical. These 10 examples were specifically chosen because they aligned with the expected outcomes based on the analogy relationships. It's important to note that this method did not work for every word tested, although it was effective for most of them.

**Question 1. 4** The section of A1\_Section1\_starter.ipynb that is labelled Bias in Word Vectors illus- trates examples of bias within word vectors in the notebook, as also discussed in class. Choose a context that you’re aware of (different from those already in the notebook), and see if you can find evidence of a bias that is built into the word vectors. Report the evidence and the conclusion you make from the evidence. [2 points]

In addition to the commonly observed gender biases (demonstrated in the notebook), the language used to train word models can also reflect socioeconomic biases inherent in our word choices. Terms associated with poorer economic status might be further from these positive descriptors and closer to words like "lazy." This illustrates a classist bias in how we frame socioeconomic status, particularly when models can reinforce stereotypes by portraying poor individuals as inherently lazy, suggesting that their struggles are a result of their own shortcomings rather than institutional or inherited disadvantages of an unfair system.

A screenshot of a computer code

Description automatically generated

Example 1.4.1: Comparative distance of wealth terms with negative connotations.

Even more troubling is when these socioeconomic biases intersect with racial lines, especially when concerning the dynamics between dominant and privileged groups and historically marginalized groups. Perhaps most troubling is the models’ apparent alignment with historical racial prejudices, particularly along lines of class and perceived intellect. This alignment may reflect the data on which the embeddings were trained, as well as the negative stereotypes present in the text overall.

A screenshot of a computer code

Description automatically generated

Example 1.4.2: Racial and socioeconomic bias within vector embeddings on an analogous basis.

**Question 1.5** Change the embedding dimension (also called the vector size) from 50 to 300 and re-run the notebook including the new cosine similarity function from part 2 above. How does the euclidean difference change between the various words in the notebook when switching from d=50 to d=300? How does the cosine similarity change? Does the ordering of nearness change? Is it clear that the larger size vectors give better results - why or why not? [5 points]

Cosine similarity appears to decrease across the examples demonstrated previously.

For instance, the cosine similarity between "apple" and "banana" decreased from 0.5608 to 0.3924. Similarly, the cosine similarity between "good" and "bad" dropped from 0.7965 to 0.6445. The sharpest decline is observed in the pair "good" and "perfect," where the similarity fell significantly from 0.8376 to 0.5893. This decline reflects a deepening of semantic isolation; while "good" and "perfect" might seem closely related on the surface, there is a substantial semantic gap between them. In contrast, "good" and "bad," though they are opposites, still share similar semantic concepts and can be compared within a broader categorical context, despite their lower similarity score.

A close-up of words

Description automatically generated

A close-up of words

Description automatically generated

Example 1.5.1: “Good” and “Perfect” difference before and after dimension adjustment.

The ordering changed dramatically as well. Words with more expansive connections, such as "nurse" or "health," showed revised lists that included terms more directly related, such as "nursing" and "nurses." Additionally, "benefits" and "care" were added, reflecting how "health" is often associated with the healthcare system in usage. Conversely, more obscure words like the names "Elizabeth" and "Michael" experienced less significant changes, with only one or two outlier adjustments. Overall, the increase in vector size appears to benefit words with more links and greater contextual relevance in usage.

A screenshot of a computer

Description automatically generated

Example 1.5.2: “Health” Adjacent words after dimension adjustment.

The ordering exercise conducted in question 1.2 also revealed significant changes. For the "dog" example, "dogs" and "cat" swapped positions in both cosine similarity and Euclidean distance rankings. This suggests that, in natural comparisons, the plural form of a word may be more similar to the singular form than other words. Other notable changes include the higher ranking of "pet" and "puppy" over less relevant terms like "horse." The appearance of "hound" in both lists reflects a deeper semantic connection due to the increased context provided by more dimensions in the word embedding vectors.

Similarly, the word "computer" saw several notable improvements. As with the previous example, the plural form "computers" rose to the top position, followed closely by "software" and the newly emergent term "pc," which is a common synonym for "computer." Additionally, the verb form "computing" was added to the list, demonstrating the various ways the base word "computer" can be used. Overall, the final ordering for "computer" is: "computers," "software," "pc," "technology," "computing." This represents a much-improved list compared to the output from word embeddings with 50 dimensions.

|  |  |  |  |
| --- | --- | --- | --- |
| Adjacent to “Dog” | Cosine Similarity | Adjacent to “Computer” | Cosine Similarity |
| Dogs | 0.79 | Computers | 0.82 |
| Cat | 0.68 | Software | 0.73 |
| Pet | 0.63 | Pc | 0.62 |
| Puppy | 0.59 | Technology | 0.62 |
| Hound | 0.55 | Computing | 0.62 |

Table 1.5: “Dog” and “Computer” adjacent words after dimension adjustment.

**Question 1.6** There are many different pre-trained embeddings available, including one that tokenizes words at a sub-word level called FastText. These pre-trained embeddedings are available from Torchtext. Modify the notebook to use the FastText embeddings. State any changes that you see in the Bias section of the notebook. [2 points]

It is apparent that FastText embeddings exhibit much less gender-based bias compared to the GloVe embeddings used previously. A notable example of this can be seen in the analogy comparisons between "men" and "doctor" and "women" and "nurse." In the GloVe example, the words "nurse," "pregnant, "child” and "mother" were used. In contrast, the FastText example employs "doctress" as the most neutral comparative word, despite its less common usage compared to "doctor."

A close-up of a computer screen

Description automatically generated

Example 1.6.1: Resulting output of “Doctor” – “Man” + “Women” using GloVe Embeddings.

A screenshot of a computer

Description automatically generated

Example 1.6.2: Resulting output of “Doctor” – “Man” + “Women” using FastText Embeddings.

This suggests that FastText shows a gender-neutral bias when assigning the profession of "doctor," as opposed to relating gender to the profession.

Similarly, the "programmer" example in FastText also yields more neutral results compared to GloVe embeddings. The list includes terms directly related to "programmer," with many words incorporating the "programmer/" prefix, indicating that gender is almost entirely irrelevant in the embedding's consideration of the term. The words are:

A screen shot of a computer

Description automatically generated

Example 1.6.3: Resulting output of “Programmer” – “Man” + “Women” using GloVe Embeddings.

A screenshot of a computer code

Description automatically generated

Example 1.6.4: Resulting output of “Programmer” – “Man” + “Women” using FastText Embeddings.

This demonstrates that FastText embeddings provide a more neutral representation of gender when it comes to professions like programming.