Sample questions and Chatbot responses

1. ***What are word embeddings used for?***

   Response:

   Word embeddings are used to represent words as vectors of numeric values. These vectors capture semantic relationships between words, meaning words that are similar in meaning will have vectors that are close together in the vector space.
   This allows for tasks like:

- **Approximating word similarity:** Words with similar meanings will have vectors closer together.
- **Disclosing hidden semantic relationships:** Embeddings can reveal relationships between words that aren't immediately obvious, such as analogies (e.g., "cat" is to "kitten" as "dog" is to "puppy").

2. ***What is Word Mover's Distance (WMD), and how does it work?***

   Response:

   Word Mover's Distance (WMD) is a metric for measuring the semantic similarity between two documents. It is based on the idea that two documents are more similar if their word embeddings are closer together in a high-dimensional space. WMD works by first generating word embeddings for each document using a pre-trained word embedding model. Then, it calculates the earth mover's distance between the two sets of word embeddings. The earth mover's distance is a metric that measures the minimum "work" required to transform one set of points into another. In the context of WMD, this means finding the minimum amount of "work" needed to transform the word embeddings of one document into the word embeddings of another document.
   A smaller WMD indicates greater semantic similarity between the two documents.

3. ***Why might high-dimensional embeddings still suffer from bias or noise?***

   Response:

   Here are some potential reasons why high-dimensional word embeddings might still suffer from bias or noise, even with higher dimensionality:

* **Data Bias:** Word embeddings are trained on text data, and if the training data itself is biased (e.g., if it over-represents certain topics or demographics), the resulting word vectors will reflect those biases. High dimensions don't automatically fix data problems.

* **Overfitting:** A model can overfit to the training data, memorizing patterns rather than learning general linguistic relationships. High dimensions can increase the risk of overfitting if not enough training data is available.

* **Noise in the data:** Even without intentional bias, raw text data can contain noise (typos, slang, etc.), which can propagate into the embeddings. More dimensions don't eliminate noise.

* **Curse of dimensionality:** In extremely high dimensions, the "curse of dimensionality" can kick in. This means that with many dimensions, data points can become sparse, making it harder to find meaningful patterns.

* **Evaluation Metrics:** Word embeddings are often evaluated on tasks like similarity tasks, which may not fully capture the nuances of word meaning. Even if embeddings are high dimensional, they might not perform well on all tasks.

It's important to note that high dimensionality is not bad in itself. Higher dimensions can be beneficial for capturing semantic

However, the provided text focuses on the CBOW and Skip-Gram models, so it

4. ***Are there limitations in using pre-trained word vectors for domain-specific tasks?***

   Response:

   While the provided text discusses word embeddings and their applications, it doesn't explicitly mention limitations of pre-trained word vectors for domain-specific tasks. Therefore, I cannot answer your question based on the given context.

5. ***Is WMD computationally expensive? Why or why not?***

   Response:

   The provided text doesn't contain information about whether or not WMD is computationally expensive.

6. ***List all types of word embeddings mentioned in the blog post.***

Response:

The blog post mentions the following types of word embeddings:

Word2Vec: This includes both the Skip-gram and Continuous Bag-of-Words (CBOW) models.
GloVe (Global Vectors for Word Representation): This method learns word embeddings from global word co-occurrence statistics.
Negative Sampling (NEG): A technique used in word2vec to approximate the NCE loss function more efficiently.
Let me know if you have any other questions.