NLP 2025 Lab 2 Report: Word Embeddings and Information Retrieval

Nichita Bulgaru
Group 52
n.bulgaru@

Timur Jercaks
Group 52
t.jercaks@

Dmitrii Sakharov
Group 52
d.sakharov@

Abstract

This report details the work undertaken for Lab 2, focusing on Word Embeddings and Information Retrieval. Utilizing a sentence compression dataset sourced from news articles, the lab explored various text representation techniques for an information retrieval task. Initial steps involved data preprocessing, including text cleaning (lowercasing, punctuation removal, lemmatization with POS tagging) and tokenization using NLTK. Subsequently, Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) vector representations were implemented and analyzed. Pre-trained GloVe word embeddings were loaded using the 'gensim' library, and their semantic properties were investigated through word similarity comparisons. A baseline method for generating sentence embeddings by averaging constituent word embeddings was implemented. These different vector representations (BoW, TF-IDF, averaged word embeddings) were then applied to the core task: retrieving original sentences given their compressed versions as queries, employing cosine similarity for ranking. The effectiveness of these retrieval methods was quantitatively evaluated using the Recall@K metric for various values of K. A comparative analysis of the representation techniques' performance on this task is presented, alongside results from experiments using an alternative pretrained model (FastText) and discussions on the strengths, weaknesses, and potential improvements for each approach.

1 Exercise 1: Clean function

The first step in preparing the sentence compression dataset for analysis involved text preprocessing. A Python function named 'clean' was implemented to normalize the text by applying several standard cleaning techniques. This function ensures consistency across the dataset, which is crucial for subsequent representation methods like Bag-of-Words (BoW) and TF-IDF, as well as for applying word embeddings effectively.

The 'clean' function performs the following preprocessing steps sequentially:

1. **Lowercasing:** All text is converted to lowercase using the string method `text.lower()`. This step standardizes the text by treating words like "Bank" and "bank" as identical, preventing them from being considered different tokens due to case variations.

2. **Punctuation Removal:** Punctuation marks are removed using regular expressions. The pattern `r"[^\w\s]"` matches any character that is not an alphanumeric character (`\w`) or whitespace (`\s`). The `re.sub()` function replaces these matched characters with an empty string, effectively stripping most punctuation while preserving word integrity.

3. **Lemmatization with Part-of-Speech (POS) Tagging:** Words are reduced to their base or dictionary form (lemma) using the `nltk.stem.WordNetLemmatizer`. To enhance the accuracy of lemmatization, especially for words that can function as different parts of speech (e.g., "running" as a verb vs. a noun), POS tagging is employed first using `nltk.pos_tag()`. A helper function, `get_wordnet_pos()`, maps the NLTK POS tags (like 'VB', 'NN') to the format required by the WordNet lemmatizer (like `wordnet.VERB`, `wordnet.NOUN`). This context-aware lemmatization provides more meaningful base forms compared to simple stemming or context-unaware lemmatization.

It is important to note that stop word removal was deliberately omitted in this initial cleaning function. As mentioned in the notebook's comments, removing common words like "is", "and", "the"

could potentially alter the semantic context, which might be disadvantageous for tasks like sentence compression that rely on preserving meaning. Furthermore, retaining stop words allows for a clearer observation of their impact (or lack thereof) in the basic BoW and TF-IDF representations explored later in the lab.

The core implementation of the ‘clean‘ function is presented below:

```
def clean(text):
    """
    Cleans the given text
    Args:
        text: a str with the text to clean

    Returns: a str with the cleaned text

    """

    # Empty text
    if text == '':
        return text

    # 'text' from the example can be of type numpy.str_, let's
 convert it to a python str
    text = str(text)

    #you might need more
    #add them here

    ### YOUR CODE HERE
    text = text.lower() # bringing to the lowercase

    # punctuation removal
    pattern = r"[^\w\s]"
    text = re.sub(pattern, "", text)

    # stop words removal (is, and, the...) is not nescessary,
may transform the context, especially when we are dealing with
 the sentence compression

    # lemmatization
    lemmatizer = nltk.stem.WordNetLemmatizer()
    tagged = nltk.pos_tag(text.split())
    text = [lemmatizer.lemmatize(n, get_wordnet_pos(pos)) for
n, pos in tagged]

    text = "␣".join(text)
    ### YOUR CODE ENDS HERE

    text = text.strip()

    # Update the example with the cleaned text
    return text
```

This function was applied consistently to both the original sentences and their compressed counterparts within the dataset using a wrapper function (‘clean_dataset‘) and the Hugging Face ‘datasets‘ library’s ‘map‘ method. This created new columns (’clean_sentence’, ’clean_compressed’) containing the processed text, ready for tokenization and subsequent analysis.

2 Exercise 2: Tokenize function

Following the cleaning step, the next crucial stage in text preparation is tokenization – the process of splitting the cleaned sentences into individual words or tokens. This is a fundamental requirement for creating count-based representations like Bag-of-Words (BoW) and TF-IDF, and also serves as the input unit when looking up word embeddings.

For this task, the standard ‘word_tokenize()‘ function from the Natural Language Toolkit (NLTK) library (`nltk.tokenize.word_tokenize`) was employed. This tokenizer is widely used and generally robust, handling common cases like contractions (though most punctuation was already removed in the ‘clean‘ step). While NLTK offers other tokenizers (e.g., ‘RegexpTokenizer‘, ‘WhitespaceTokenizer‘), ‘word_tokenize‘ provides a good balance of sophistication and simplicity for this dataset.

The implementation of the ‘tokenize‘ function is straightforward:

```
def tokenize(text):
    """
    Tokenizes the `text` parameter using nltk library
    Args:
        text: a string representing a sentence to be tokenized

    Returns: a list of tokens (strings)

    """

    ### YOUR CODE HERE

    tokens = nltk.tokenize.word_tokenize(text)

    ### YOUR CODE ENDS HERE
    return tokens
```

Similar to the cleaning process, a wrapper function (‘tokenize_dataset‘) was created to apply the ‘tokenize‘ function to the ’clean_sentence’ and ’clean_compressed’ columns of the dataset. The ‘map‘ method efficiently processed the entire dataset, adding two new columns: ’sentence_tokens’ and ’compressed_tokens’. Each entry in these columns is a list of strings, where each string represents a token from the corresponding cleaned sentence. This tokenized format is essential for the subsequent steps involving vocabulary creation and vector representation.

3 Exercise 3: Extracting vocabulary counts

To construct the Bag-of-Words (BoW) and TF-IDF representations, a vocabulary must first be built from the corpus. This involves identifying all unique tokens and, typically, their frequencies. Exercise 3 focused on implementing a function to count these frequencies.

The function ‘extract_vocabulary_counts‘ takes a list of tokenized sentences (where each sentence is itself a list of tokens) as input. It utilizes the ‘collections.Counter‘ class, which is highly efficient

for frequency counting tasks. The function iterates through each tokenized sentence in the input list and uses the ‘update()‘ method of the ‘Counter‘ object to increment the count for each token encountered.

The implementation is as follows:

```
def extract_vocabulary_counts(tokenized_sentences):
    """
    Extracts the vocabulary from the tokenized
    sentences
    Args:
        tokenized_sentences: a list of lists of tokens

    Returns: a Counter object with the counts of
    each word in the vocabulary
    """

    ### YOUR CODE HERE

    counter = Counter()
    for n in tokenized_sentences:
        counter.update(n)

    return counter

    ### YOUR CODE ENDS HERE
```

This function was applied to the combined list of tokenized original sentences and tokenized compressed sentences from the *training* split of the dataset. This ensures the vocabulary captures words present in both forms of the text.

```
# Extracting tokens from the training split
tokenized_sentences = split_ds['train']['sentence_tokens']
tokenized_compressed = split_ds['train']['compressed_tokens']

# Combining and counting
vocab_counter = extract_vocabulary_counts(
                    tokenized_sentences + tokenized_compressed
)

print(len(vocab_counter))
# Output: 109366

print(vocab_counter.most_common(10))
# Output: [('the', 182927), ('a', 135981), ('to', 130228),
#          ('in', 112990), ('of', 100233), ('be', 93623),
#          ('and', 68179), ('on', 56280), ('have', 55965),
#          ('for', 52007)]
```

The resulting vocabulary contains 109,366 unique tokens. As observed from the 10 most common tokens, the highest frequency words are predominantly stop words (e.g., ’the’, ’a’, ’to’, ’in’, ’of’) and common verbs (’be’, ’have’). This large vocabulary size, heavily influenced by rare words (as seen in Lab 1), necessitates limiting the vocabulary for practical BoW/TF-IDF representations, which is addressed in subsequent steps.

4 Exercise 4: Bag of Words

The Bag-of-Words (BoW) model is a fundamental technique for representing text documents as numerical vectors. It disregards grammar and word order but keeps track of word multiplicity. Exercise 4 required implementing a function to generate the BoW vector for a single sentence, given a limited vocabulary.

Before implementing the BoW function itself, the full vocabulary derived in Exercise 3 (containing 109,366 unique tokens) was reduced. Only the 10,000 most frequent tokens were retained, which is a common practice to manage dimensionality and focus on more informative words. A dictionary, ‘token_to_id‘, was created to map each of these top 10,000 tokens to a unique integer index (from 0 to 9999).

The ‘bag_of_words‘ function takes two arguments: a sentence represented as a list of tokens and the ‘token_to_id‘ mapping. It returns a NumPy array of size equal to the vocabulary size (10,000 in this case). The function works as follows:

1. Initializes a NumPy array (‘bow‘) of zeros with a length equal to the vocabulary size.
2. Iterates through each token in the input sentence list.
3. For each token, it checks if the token exists as a key in the ‘token_to_id‘ dictionary (i.e., if it’s part of the limited vocabulary).
4. If the token is in the vocabulary, its corresponding index is retrieved from ‘token_to_id‘.
5. The count at that specific index in the ‘bow‘ array is incremented by one.
6. Tokens not present in the ‘token_to_id‘ map (Out-Of-Vocabulary words relative to the top 10k) are ignored.
7. Finally, the resulting ‘bow‘ array, representing the frequency counts of vocabulary words in the sentence, is returned.

The Python implementation is shown below:

```
def bag_of_words(sentence, token_to_id):
    """
    Creates a bag-of-words representation of the sentence
    Args:
        sentence: a list of tokens
        token_to_id: a dictionary mapping each word
        to an index in the vocabulary

    Returns:: a numpy array of size vocab_size with
    the counts of each word in the vocabulary

    """
    vocab_size = len(token_to_id)
    bow = np.zeros(vocab_size, dtype=int)

    ### YOUR CODE HERE

    for n in sentence:
        if n in token_to_id.keys():
```

```
            id = token_to_id[n]
            bow[id]+=1

    ### YOUR CODE ENDS HERE

    return bow
```

This function provides a simple yet effective way to convert a sequence of tokens into a fixed-size numerical vector based on word counts. The notebook demonstrates its application on a sample sentence, showing the resulting sparse integer vector. This vector representation is the foundation for comparing sentences using cosine similarity in the BoW-based retrieval system developed later.

5 Exercise 5: Cosine Similarity between two vectors

To compare the vector representations of sentences (whether BoW, TF-IDF, or embeddings), a similarity measure is required. Cosine similarity is a widely used metric in information retrieval and NLP for this purpose. It measures the cosine of the angle between two non-zero vectors in a multi-dimensional space, effectively capturing their orientation similarity irrespective of their magnitude. A cosine similarity value of 1 indicates identical direction, 0 indicates orthogonality (no similarity in direction), and -1 indicates opposite directions.

Exercise 5 involved implementing a function ‘cosine_similarity‘ that takes two NumPy arrays (representing the vectors) as input and returns their cosine similarity score.

The mathematical formula for cosine similarity between two vectors $\mathbf{a}$ and $\mathbf{b}$ is:

$$\begin{aligned} \text{similarity} = \cos(\theta) &= \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \\ &= \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \end{aligned} \tag{1}$$

where $\mathbf{a} \cdot \mathbf{b}$ is the dot product of the vectors, and $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are their Euclidean norms (magnitudes).

The implementation leverages NumPy’s optimized functions for efficiency:

1. Calculates the Euclidean norm (L2 norm) of each input vector using `np.linalg.norm()`.
2. Calculates the dot product of the two vectors using `np.dot()`.
3. Divides the dot product by the product of the norms to get the cosine similarity.
4. An alternative, often computationally equivalent for unit vectors, is to normalize each vector first (divide by its norm) and then compute their dot product. The provided solution uses this approach implicitly by calculating norms separately.

The implemented Python function is:

```
def cosine_similarity(vector1, vector2):
    """
    Computes the cosine similarity between two vectors
    Args:
        vector1: numpy array of the first vector
        vector2: numpy array of the second vector

    Returns: cosine similarity

    """
    ### YOUR CODE HERE
    norm_1 = np.linalg.norm(vector1)
    normalized_vec_1 = vector1 / norm_1 if norm_1 != 0 else
vector1

    norm_2 = np.linalg.norm(vector2)
    normalized_vec_2 = vector2 / norm_2 if norm_2 != 0 else
vector2

    # Handle potential zero vectors to avoid division by zero
/ NaN
    if norm_1 == 0 or norm_2 == 0:
        return 0.0 # Or handle as appropriate (e.g., NaN,
error)

    similarity = np.dot(normalized_vec_1, normalized_vec_2)

    return similarity
    ### YOUR CODE ENDS HERE
```

This function serves as a core component for the retrieval system, allowing the similarity between a query vector and all sentence vectors in the dataset to be calculated efficiently. The notebook includes examples demonstrating its usage with simple vectors.

6 Exercise 6: Cosine Similarity between a vector and an array of vectors

While the function from Exercise 5 calculates the similarity between two individual vectors, a typical information retrieval scenario involves comparing a single query vector against a large collection of document (or sentence) vectors. Performing pairwise comparisons using the ‘cosine_similarity‘ function in a loop would be inefficient.

Exercise 6 required implementing a more optimized function, ‘cosine_similarity_1_to_n‘, to compute the cosine similarities between one query vector and *all* vectors stored in a 2D NumPy array (matrix). The function takes the query vector (1D array) and the matrix of other vectors (2D array of size $N \times D$, where N is the number of sentences and D is the vector dimension) as input, and returns a 1D array of size N containing the similarity scores.

1. **Dot Product:** The dot product between the query vector ($\mathbf{q}$, size D) and the matrix of sentence vectors ($\mathbf{S}$, size $N \times D$) is computed. To get an N-dimensional result vector where each element is $\mathbf{q} \cdot \mathbf{s}_i$, we compute $\mathbf{S} \cdot \mathbf{q}^T$ (matrix-vector multiplication, treating $\mathbf{q}^T$ as a column vector). NumPy's `np.dot(other_vectors, vector)` achieves this efficiently. The result is a 1D array of size N.

2. **Norms:**
   - The norm of the single query vector ($\|\mathbf{q}\|$) is calculated using `np.linalg.norm(vector)`.
   - The norms of all sentence vectors ($\|\mathbf{s}_i\|$) are calculated simultaneously using `np.linalg.norm(other_vectors, axis=1)`. The 'axis=1' argument ensures the norm is computed across the columns (dimension D) for each row (sentence). This yields a 1D array of size N.

3. **Final Calculation:** The resulting dot product array (size N) is divided element-wise by the product of the query vector's norm (a scalar) and the array of sentence vector norms (size N). NumPy handles the broadcasting of the scalar norm appropriately.

The implemented function is:

```
def cosine_similarity_1_to_n(vector, other_vectors):
    """
    Calculates the cosine similarity between a
    single vector and other vectors.
    Args:
        vector: a numpy array representing a vector
        of D dimensions
        other_vectors: a 2D numpy array representing
        other vectors
        (of the size NxD, where N is the
        number of vectors and D is their dimension)

    Returns: a 1D numpy array of size N containing
    the cosine similarity between
    the vector and all the other vectors

    """

    #### YOUR CODE HERE
    dot_product = np.dot(other_vectors, vector) #
    1xN vector

    norms_others = np.linalg.norm(other_vectors, axis=1)
    norm_vector = np.linalg.norm(vector)

    zero_norms_mask = (norm_vector == 0) |
    (norms_others == 0)
    similarities_vector =
    np.zeros_like(dot_product, dtype=float)

    valid_mask = ~zero_norms_mask
    denominator = (norm_vector * norms_others[valid_mask])
    if np.any(denominator): # Ensure denominator is
    not zero
        similarities_vector[valid_mask] = (
            dot_product[valid_mask] / denominator
        )
        # Ensure results are within [-1, 1] due to
        potential float errors
        np.clip(similarities_vector, -1.0, 1.0,
        out=similarities_vector)

    return similarities_vector

    ### YOUR CODE ENDS HERE
```

This function is significantly faster for comparing one query against many sentences than iterating with the previous function, making it suitable for the retrieval evaluation step. The notebook demonstrates its use by calculating the similarity between a sample query's BoW vector and all sentence BoW vectors in the test set.

7 Exercise 7: Analyzing and improving BOW search results

This exercise involved a critical analysis of the baseline Bag-of-Words (BoW) information retrieval system, followed by the implementation and evaluation of specific improvements to address its observed weaknesses.

7.1 Performance Analysis of Basic BoW Search

The initial BoW retrieval system utilized BoW vectors derived from the top 10,000 most frequent tokens (token_to_id) without stop word removal, using cosine similarity for ranking via the perform_search function. Analysis of search results for various queries revealed significant limitations inherent to this simple representation: **Failure Cases and Examples (as observed experimentally):**

1. **Domination by Common/Stop Words:** Queries containing high-frequency words like 'and' often retrieved irrelevant documents dominated by these terms.
   - *Query:* 'fox and deer'
   - *Result:* Top results were unrelated financial texts containing 'and' frequently.

2. **Semantic/Synonym Insensitivity:** The model failed on semantic nuances, unable to link synonyms or related concepts.
   - *Query:* 'Chinese planes'
   - *Result:* Retrieved general "Chinese" documents, missing the "aviation" context.
   - *Query:* 'reply' vs. 'respond'

- *Result:* Search for 'reply' missed relevant sentences containing 'respond'.

3. **TF Imbalance:** Sentences with high counts of a single query term could outrank sentences matching multiple, rarer query terms.
   - *Query:* 'asian Elephant'
   - *Result:* A sentence containing only "elephant" ranked higher than more specific sentences with both terms.
4. **Lack of Phrase Understanding:** Multi-word concepts were treated as independent words.
   - *Query:* 'blood pressure'
   - *Result:* Retrieved documents containing either 'blood' or 'pressure' individually.
5. **Short Sentence Bias:** Cosine similarity sometimes favored shorter sentences for short queries.
   - *Query:* 'Nicole'
   - *Result:* Shorter mentions ranked higher than potentially more informative longer articles.
6. **Out-of-Vocabulary (OOV) Words:** Queries with terms outside the top 10k vocabulary yielded poor results.
   - *Query:* 'preventive' (Observed to be OOV or very rare in the initial 10k vocab)
   - *Result:* Irrelevant results with near-zero similarity.

Success Cases (Observed): The baseline performed adequately when queries consisted of specific, relatively unique terms present in the vocabulary.

- *Query:* 'Zosel Dam Lake Osoyoos'
- *Result:* The exact matching sentence was retrieved as the top result.

7.2 Strategy for Improving BoW Search

Acknowledging the inherent semantic limitations of BoW, the following improvements were implemented to address the observed practical issues:

1. **Stop Word Removal:** NLTK's standard English stop word list was applied to filter tokens before vectorization, aiming to reduce noise from common words. The filtering logic was integrated into the preprocessing pipeline feeding the improved search function.

```
from nltk.corpus import stopwords

stops = set(stopwords.words('english'))
stop_words = set(stopwords.words('english'))
print(stops)
# in case it is not there. As we will use stop
words deletion on already lemmatized words, we
might need that
stops.add("n't")
```

2. **N-grams (Bigrams):** To capture simple word co-occurrences and improve phrase matching, bigrams (word pairs) were generated from the stopword-removed tokens and added to the vocabulary alongside unigrams.

```
def generate_ngrams(tokens, n_values=[1, 2]):
    all_ngrams_list = []
    for n in n_values:
        if n == 1:
            all_ngrams_list.extend(tokens)
        elif len(tokens) >= n:
            n_gram_tuples = nltk.ngrams(tokens, n)
            n_gram_strings = ["_".join(gram) for
            gram in n_gram_tuples]
            all_ngrams_list.extend(n_gram_strings)
    return all_ngrams_list
```

3. **Increased Vocabulary Size:** The vocabulary size was doubled to 20,000 terms (vocab_size = 20000) to accommodate unigrams and bigrams and reduce the likelihood of OOV query terms. A new token_to_id_ngrams mapping was created based on this larger vocabulary.

```
vocab_size = 20000
vocab_ngrams = vocab_counter_ngrams.most_common(
vocab_size)
token_to_id_ngrams = {term: i for i, (term, _) in
enumerate(vocab_ngrams)}
example_bigram_in_vocab = [item for item in
token_to_id_ngrams if '_' in item]
```

4. **Length Weighting Factor:** To mitigate the bias towards shorter sentences, a length-based bonus was added during the ranking phase within the perform_search_factor function. A small weight (weight_factor, default 0.001) multiplied by the logarithm of the sentence length (np.log(sentence_length + 1)) was added to the initial cosine similarity score before final sorting. This gives a slight preference to longer documents without drastically altering the core similarity ranking.

```
def perform_search_factor(query, num_results=5,
weight_factor=0.001):

    embedded_query = get_query_embedding_ngram(query,
token_to_id_ngrams)


    query_similarity = cosine_similarity_1_to_n(
embedded_query, sentences_bows_ngrams)
```

```
    query_similarity = np.nan_to_num(query_similarity)

    candidate_k = max(num_results * 2, 20)
    initial_top_indices = top_k_indices(query_similarity
, k=candidate_k, sorted=True).tolist()

    scored_candidates = []
    for idx in initial_top_indices:
        original_similarity = query_similarity[idx]

        sentence_length = len(split_ds['test'][idx]['
sentence_tokens'])
        length_bonus = weight_factor * np.log(
sentence_length + 1)

        adjusted_score = original_similarity +
length_bonus
        scored_candidates.append((adjusted_score,
original_similarity, idx, sentence_length))

    scored_candidates.sort(key=lambda x: x[0], reverse=
True)


    for adjusted_score, original_similarity, idx,
sentence_length in scored_candidates[:num_results]:

        text = split_ds['test'][idx]['set'][0]

        print(text)
```

The improved search logic, incorporating stop word removal, n-gram generation, the larger vocabulary, and BoW vector creation, was encapsulated primarily within the get_query_embedding_ngram function, which was then called by perform_search_factor. The length weighting was applied after retrieving initial candidates based on cosine similarity in perform_search_factor.

7.3 Impact of Improvements

The implemented improvements were tested using the perform_search_factor function, demonstrating the following effects:

- **OOV Handling:** The query 'preventive' successfully retrieved relevant sentences, confirming the benefit of the larger vocabulary (Improvement 3).

```
Despite preventive steps being taken by the authorities,
 poaching continues in Jim Corbett National Park in
Uttarakhand''s Ramnagar district.
Tamoxifen, taken by certain women as a preventive
measure against breast cancer, saves lives and reduces
medical costs.
Spanish tenor Placido Domingo is to undergo ``preventive
 surgery'' and will be unable to perform for the next
six weeks, his spokesman has said.
Spanish tenor Placido Domingo is to undergo ``preventive
 surgery'' and will be unable to perform for the next
six weeks, his spokesperson has said.
There is not enough evidence to support suicide
screening for all teens and adults, according to the US
Preventive Services Task Force.
```

- **Phrase Handling:** The query 'blood pressure' showed improved results, retrieving sentences containing the specific phrase more accurately due to the inclusion of bigrams like "blood_pressure" in the vocabulary (Improvement 2).

```
Free blood pressure checks will be on offer around Otago
 today to encourage people to find out if they are at
risk of having a stroke.
A new study has found that exposure to secondhand smoke
raises blood pressure in boys, but not in girls.
After his performance on American Idol Wednesday night,
contestant David Cook was rushed to the hospital after
suffering from high blood pressure and experiencing
heart palpitations.
Blood For Blood is reuniting and heading out on tour
after six years.
Electric Boat gave $125,000 on Friday to 149 employees
who took part in a company program that encourages them
to quit smoking, get a physical, check health numbers
like blood pressure or use the Electric Boat pharmacy.
```

- **Length Bias Mitigation:** The query 'Nicole' resulted in a better ranking distribution, with longer, relevant articles appearing higher in the results when a suitable weight_factor (e.g., 0.1) was applied, demonstrating the effectiveness of the length bonus (Improvement 4).

```
Nicole Scherzinger wants a big family Updated: 04:58,
Sunday September 2, 2012 Nicole Scherzinger hopes to
have lots of kids in the future and is looking to Gary
Barlow, her co-star on 'The_X_Factor' for advice.
Nicole Scherzinger, the lead singer of the Pussycat
Dolls, says she was a shy child and had a strict
upbringing.
Oscar winning actress Nicole Kidman admits that she is '
boring' as she is a teetotaller.
Anna Nicole Smith was so weak in the days before her
death that she used a baby bottle to get fluids and
drifted in and out of consciousness, an investigator
testified Tuesday.
Actor Dul Hill, best known for his work on West Wing and
 his current gig on Psych, has filed for separation from
 his wife, actress Nicole Lyn.
```

- **Stop Word Effect:** The removal of stop words (Improvement 1) inherently prevented queries like 'fox and deer' from being skewed by the high frequency of "and", leading to more topically relevant (though not necessarily perfect) results focused on "fox" or "deer". While not explicitly re-run with perform_search_factor in the final notebook state provided for this query, the removal of stop words is a direct countermeasure to the originally observed failure mode.

- **Limitations:** Despite improvements, the fundamental inability to grasp deep semantics persisted. Queries like 'Chinese planes' likely benefited slightly from bigrams (e.g., "Chinese_premier") but still struggled with the broader concept compared to embedding methods. Synonymy (reply/respond) remained unaddressed. The basic TF weighting issue seen with 'asian Elephant' was also not resolved by these specific BoW modifications.

In summary, the implemented improvements demonstrably addressed several key weaknesses of the baseline BoW system related to stop words, simple phrases, OOV terms, and length bias. The system became more robust for queries relying on these aspects. However, for tasks requiring deeper semantic understanding or more nuanced term weighting, limitations inherent to the BoW model remained apparent, suggesting the need for more advanced techniques like TF-IDF or embeddings.

8 Exercise 8: Inverse Document Frequency (IDF)

To improve upon the plain Bag-of-Words representation, we next compute the Inverse Document Frequency (IDF) component of TF–IDF. Let N be the total number of documents (sentences) in our collection, and let $\mathrm{df}(t)$ be the number of documents in which term t appears. We apply Laplace smoothing to avoid division by zero. A common formulation is:

$$\mathrm{IDF}(t) \;=\; \log_{10}\!\left(\frac{N}{\mathrm{df}(t)+1}\right).$$

Below is the Python implementation, using our existing BoW matrix of shape $(N \times D)$, where D is the vocabulary size.

```
def calculate_idf(bows):
    """
    Calculates the IDF for each word in the vocabulary
    Args:
        bows: numpty array of size (N x D) where N is the
number of documents and D is the vocabulary size

    Returns: a numpy array of size D with IDF values for each
token
    """

    ### YOUR CODE HERE

    size = bows.shape

    df = np.count_nonzero(bows > 0, axis=0)+1
    idfs = np.log10(size[0]/df)


    return idfs
    ### YOUR CODE ENDS HERE
```

This yields a length-D vector of IDF weights, which we will multiply elementwise with our BoW counts to obtain the TF–IDF representations in the next exercise.

9 Exercise 9: TF–IDF

Term Frequency–Inverse Document Frequency (TF–IDF) combines the raw term frequencies from our BoW vectors with the IDF weights computed in Exercise 8:

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t),$$

Since our "TF" is simply the count in the BoW matrix, we obtain the TF–IDF representation by element-wise multiplying each BoW row by the IDF vector.

9.1 Computing TF–IDF Matrices

```
idf_train = calculate_idf(sentences_bows)
print(f"IDF vector shape: {idf_train.shape}")

Output: IDF vector shape: (10000,)

sentences_bows_test = test_ds['sentence_bow']
compressed_bows_test = test_ds['compressed_bow']
print(f"Test set BOW shape: {sentences_bows_test.shape}")

Output: Test set BOW shape: (36000, 10000)

sentences_tfidf_test = sentences_bows_test * idf_train
compressed_tfidf_test = compressed_bows_test * idf_train

print(f"Test set TF-IDF shape: {sentences_tfidf_test.shape}")

Output: Test set TF-IDF shape: (36000, 10000)
```

After this step, `tfidf_test` and `compressed_tfidf_test` contain the TF–IDF vectors for each sentence in the test set.

9.2 TF–IDF Search Function

We implement two helper functions: one to embed a query into TF–IDF space, and one to perform retrieval via cosine similarity.

```
def embed_query_tfidf(query_text, clean_fn, tokenize_fn,
token_to_id_map, idf_vector):

    cleaned = clean_fn(query_text)
    tokens = tokenize_fn(cleaned)
    query_bow = bag_of_words(tokens, token_to_id_map)
    query_tfidf = query_bow * idf_vector
    return query_tfidf

def perform_search_tfidf(query, num_results=5):

    embedded_query_tfidf = embed_query_tfidf(query, clean,
tokenize, token_to_id, idf)
    query_similarity_tfidf = cosine_similarity_1_to_n(
embedded_query_tfidf, sentences_tfidf_test)
    query_similarity_tfidf = np.nan_to_num(
query_similarity_tfidf)

    top_indices_tfidf = top_k_indices(query_similarity_tfidf,
k=num_results).tolist()

    for idx in top_indices_tfidf:
        similarity_score = query_similarity_tfidf[idx]
        original_sentence = split_ds['test'][idx]['set'][0]
        print(f"{original_sentence}")
```

9.3 Comparative Observations

Running TF–IDF retrieval alongside the baseline BoW retrieval shows:

- **Stop-word queries ("fox and deer")** TF–IDF down-weights "and", so results focus on "fox"

and “deer” rather than irrelevant “and”-heavy sentences.

```
    BOW
CW INDUSTRIES High density rectangular connector with
enhanced retention and solderability, meets the
requirements of MIL-DTL-28804 and NAVSEA 3164341 and
3164342 specifications.
``The Sensex has supports at 17,520 and 17,380 and
resistances at 17,710 and 17,830. The Nifty has supports
 at 5,300 and 5,270 and resistances at 5,370 and
5,400,`` it added.
Heide and Cook Ltd. is restructuring its business to
focus on heating, ventilation and air conditioning
services, and CFO and Executive Vice President Dexter
Kekua is retiring and will not be replaced.
Singapore and Australia reaffirmed on Monday to enhance
bilateral ties and strengthen cooperation on regional
and international political, economic and security
developments.
Chinese ambassador to Australia Zhang Junsai on Tuesday
called for broader and further cooperation between China
 and Australia in such areas as energy and resources,
education and tourism, and in environmental protection,
particularly in the time of the global financial crisis
and economic downturn.
TF-IDF
What time of the day is best for hunting deer during a
full moon?
Fox has apologized to Asians after a Fox Sports report
made fun of Asian students' accents and knowledge of
college football.
Authorities say a driver in west Michigan has died after
 hitting a deer, losing control of the vehicle and
striking a tree.
Divers found the body of a missing deer hunter, near
Laurie, in Camden County.
It's deer hunting season, and that means some lucky
hunters will bring home venison.
```

- **Rarer term emphasis (“Chinese planes”)** TF–IDF gives more weight to the moderately rare term “plane”, yielding better aviation-related hits, though true semantic matching still requires embeddings.

```
    BOW
Charlotte and Port Charlotte high school students
enrolled in Liang Wang's Chinese class celebrate the
Chinese New Year.
Shanghai-based Chinese steel giant Baosteel signs a
cooperation contract with Hebei-based Chinese automobile
 maker Great Wall Motor Company Limited, according to
media reports.
An interaction program on Chinese publication was held
Friday in Nepali capital Kathmandu to mark the coming
Chinese New Year.
AP , 06.08.11, 08:25 AM EDT Taiwanese and Chinese
negotiators says Taiwan will let Chinese tourists travel
 alone on the island starting in late June.
Chinese Premier Wen Jiabao will visit Brunei Darussalam
next month, the first by the Chinese premier to the
Sultanate.
TF-IDF
The plane crashed into a home in High Point.
A Greensboro man died in a small plane crash at the
Elkin Municipal Airport, not far from where six people
died in a plane crash last week.
A small plane crashed this weekend into Lake Lucille
near Wasilla and sank.
Chinese Premier Wen Jiabao will visit Brunei Darussalam
next month, the first by the Chinese premier to the
Sultanate.
Four-time Olympic champion Guo Jingjing will not retire
after the Chinese 11th National Games, Zhou Jihong, team
 leader of the Chinese national diving team, said on
Friday.
    
```

- **Over-emphasis caution (“china airlines”)** A document repeating “airlines” may outrank the exact match “China Eastern Airlines,” illustrating that TF–IDF can still over-reward frequency of mid-IDF terms.

```
    BOW
Qantas has significantly expanded its reach into China
by signing a new codeshare agreement with China Eastern
Airlines.
AirMedia Group Inc, an operator of out-of-home
advertising platforms in China, has renewed its
concession rights contract with China Eastern Airlines,
the company reported today.
For the seventh time LAN Airlines is chosen Best Airline
 in Central/ South America and the Caribbean by the
Official Airline Guide.
Starbucks Corporation will soon face more competition in
 China, said China Daily today.
A Nationwide computer failure has drastically disrupted
US carrier United Airlines operations, the airline
confirmed.
TF-IDF
For the seventh time LAN Airlines is chosen Best Airline
 in Central/ South America and the Caribbean by the
Official Airline Guide.
Qantas has significantly expanded its reach into China
by signing a new codeshare agreement with China Eastern
Airlines.
AirMedia Group Inc, an operator of out-of-home
advertising platforms in China, has renewed its
concession rights contract with China Eastern Airlines,
the company reported today.
A Nationwide computer failure has drastically disrupted
US carrier United Airlines operations, the airline
confirmed.
Aloha Airlines halting passenger service after
bankruptcy filing The airline says it will stop taking
flight reservations after today.
```

- **Synonym and phrase limitations** Like BoW, TF–IDF cannot bridge “reply” $\leftrightarrow$ “respond” or intrinsically treat “blood pressure” as a unit unless n-grams are added.

```
    reply
    BOW
WordPress announced that WordPress.com blogs are now
getting comment notifications, with moderation and reply
 functionality.
Congress president Sonia Gandhi warned on Wednesday that
 India would give a ``fitting reply'' to terrorism.
Action against S Sreesanth for criticising the Kerala
Cricket Association will depend on his reply to the
association's letter seeking the player's explanation.
Kolkata, June 8 Mohun Bagan Monday suspended soccer star
 Bhaichung Bhutia for six months, alleging that the
striker has given an 'unsatisfactory' reply to the show
cause notice slapped on him by the I-League club.
BSNL has requested a refund of Rs 8,313.9 crore for
surrendering its BWA spectrum in 20 circles, minister of
 state for communications and IT Milind Deora said in
his written reply to Lok Sabha.
Madhya Pradesh Veteran Congress leader Arjun Singh who
was Madhya Pradesh Chief Minister at the time of the
Bhopal gas disaster in December 1984 said here today
that he would give ``appropriate reply at an appropriate
 time'' on it.
In a written reply to the House, Minister of State for
External Affairs E. Ahmad said: ``India is committed to
resolving all outstanding issues with Pakistan through
dialogue in the interest of peace and prosperity of our
people,'' The Dawn reports.
The Finance Ministry is going to reply to the Vodafone
notice on the retrospective tax proposal after the
passage of the Finance Bill in Parliament as the matter
is related to it.
The former chairman of Andhra Pradesh Industrial
Infrastructure Corporation, Mr Ambati Rambabu, said the
former MP, Mr YS Jagan Mohan Reddy, will give reply to
```

```
the notices given by income tax department and alleged
that the UPA chairperson, Ms Sonia Gandhi, the Chief
Minister, Mr Kiran Kumar Reddy.
Sports Writer Adding a big arm to their bullpen, the NL
West-leading Los Angeles Dodgers acquired former All-
Star closer George Sherrill from the Baltimore Orioles
on Thursday for two minor leaguers.
TF-IDF
WordPress announced that WordPress.com blogs are now
getting comment notifications, with moderation and reply
 functionality.
Congress president Sonia Gandhi warned on Wednesday that
 India would give a ``fitting reply'' to terrorism.
Action against S Sreesanth for criticising the Kerala
Cricket Association will depend on his reply to the
association's␣letter␣seeking␣the␣player's explanation.
Kolkata, June 8 Mohun Bagan Monday suspended soccer star
 Bhaichung Bhutia for six months, alleging that the
striker has given an 'unsatisfactory' reply to the show
cause notice slapped on him by the I-League club.
The Finance Ministry is going to reply to the Vodafone
notice on the retrospective tax proposal after the
passage of the Finance Bill in Parliament as the matter
is related to it.
BSNL has requested a refund of Rs 8,313.9 crore for
surrendering its BWA spectrum in 20 circles, minister of
 state for communications and IT Milind Deora said in
his written reply to Lok Sabha.
In a written reply to the House, Minister of State for
External Affairs E. Ahmad said: ``India is committed to
resolving all outstanding issues with Pakistan through
dialogue in the interest of peace and prosperity of our
people,'' The Dawn reports.
Madhya Pradesh Veteran Congress leader Arjun Singh who
was Madhya Pradesh Chief Minister at the time of the
Bhopal gas disaster in December 1984 said here today
that he would give ``appropriate reply at an appropriate
 time'' on it.
The former chairman of Andhra Pradesh Industrial
Infrastructure Corporation, Mr Ambati Rambabu, said the
former MP, Mr YS Jagan Mohan Reddy, will give reply to
the notices given by income tax department and alleged
that the UPA chairperson, Ms Sonia Gandhi, the Chief
Minister, Mr Kiran Kumar Reddy.
The Samsung Galaxy S4 will be available to pre-order
from TalkTalk from Monday, ready for its release on
April 26.

respond
BOW
Firefighters already stretched thin in responding to
multiple fires this week, responded to a fire off of
County Road 397 Wednesday afternoon.
Police respond to a stabbing around 8 pm Monday in Terre
 Haute.
UFC President Dana White responded harshly when asked
about the crowd booing tonight's␣flyweight␣title␣fight.
The␣Red␣Cross␣is␣expanding␣its␣resources␣in␣Missouri␣to␣
respond␣faster␣during␣local␣disasters.
Crews␣responded␣to␣the␣scene␣of␣a␣natural␣gas␣leak␣in␣
Chester␣County␣on␣Thursday␣evening.
TF-IDF
Firefighters␣already␣stretched␣thin␣in␣responding␣to␣
multiple␣fires␣this␣week,␣responded␣to␣a␣fire␣off␣of␣
County␣Road␣397␣Wednesday␣afternoon.
A␣Hofstra␣student␣was␣killed␣by␣police,␣who␣were␣
responding␣to␣reports␣of␣a␣home␣invasion.
Police␣respond␣to␣a␣stabbing␣around␣8␣pm␣Monday␣in␣Terre
␣Haute.
A␣Bismarck␣Police␣officer␣is␣shot␣and␣killed␣after␣
responding␣to␣a␣call␣last␣night...
Lauryn␣Hill␣has␣turned␣to␣the␣Internet␣to␣respond␣to␣
federal␣tax␣evasion␣charges.
␣␣␣␣
```

- **OOV terms (“preventive”)** TF–IDF yields no improvement for words outside the vocabulary, mirroring BoW’s OOV failure.

```
    BOW
    Two-time world champion Fernando Alonso on Saturday
boosted his hopes of claiming his third victory at the
Monaco Grand Prix when he topped the times in final free
 practice for Sunday's␣race.
␣␣␣␣Barclays␣is␣launching␣two␣new␣savings␣accounts␣
including␣a␣best␣buy␣cash␣ISA␣paying␣3.55%␣gross/3.61%␣
AER␣interest␣to␣help␣customers␣make␣the␣most␣of␣their␣
tax-free␣savings␣allowance.
␣␣␣␣A␣minor␣died␣in␣a␣road␣accident␣at␣Dulegauda␣in␣
Tanahu␣district␣on␣Tuesday.
␣␣␣␣The␣Samsung␣Galaxy␣S4␣will␣be␣available␣to␣pre-order
␣from␣TalkTalk␣from␣Monday,␣ready␣for␣its␣release␣on␣
April␣26.
␣␣␣␣Actress␣Evelyn␣Keyes,␣who␣played␣Scarlett␣O'Hara's␣
younger␣sister␣in␣``Gone␣With␣the␣Wind,''␣has␣died␣at␣
age␣91.
␣␣␣␣TF-IDF
␣␣␣␣Two-time␣world␣champion␣Fernando␣Alonso␣on␣Saturday␣
boosted␣his␣hopes␣of␣claiming␣his␣third␣victory␣at␣the␣
Monaco␣Grand␣Prix␣when␣he␣topped␣the␣times␣in␣final␣free
␣practice␣for␣Sunday's race.
    Barclays is launching two new savings accounts
including a best buy cash ISA paying 3.55% gross/3.61%
AER interest to help customers make the most of their
tax-free savings allowance.
    A minor died in a road accident at Dulegauda in
Tanahu district on Tuesday.
    The Samsung Galaxy S4 will be available to pre-order
 from TalkTalk from Monday, ready for its release on
April 26.
    Actress Evelyn Keyes, who played Scarlett O'Hara's
younger sister in ``Gone With the Wind,'' has died at
age 91.
```

In summary, TF–IDF meaningfully improves over plain BoW by suppressing extremely common words and highlighting discriminative terms, but it retains all lexical overlap limitations. Embedding-based methods, explored in subsequent exercises, are required to address deep semantic matching.

10 Exercise 10: Plotting similarities between words

Word embeddings, such as the GloVe model loaded previously (`glove-wiki-gigaword-100`), aim to capture semantic relationships based on word co-occurrence patterns in large corpora. This exercise explores these relationships by computing and visualizing the cosine similarities between the embedding vectors of selected sets of words.

10.1 Calculating the Similarity Matrix

To facilitate the comparison, a helper function `calculate_similarity_table` was implemented. This function takes a list of words and the pre-trained embedding model as input. It iterates through all pairs of words in the list, retrieves their respective embedding vectors from the model, computes the cosine similarity between each pair using the function from Exercise 5, and stores these values in an $N \times N$ NumPy array, where N is the number of words in the list.

The implementation is as follows:

```
def calculate_similarity_table(list_words, model):
    similarity_matrix = np.zeros((len(list_words),
```

```
                                len(list_words)), dtype=
float)

    for index, n in enumerate(list_words):
        for index2, j in enumerate(list_words):
            # Assume words are in model vocabulary
            vec1 = model[n]
            vec2 = model[j]

            similarity = cosine_similarity(vec1, vec2)

            similarity_matrix[index][index2] = similarity
    return similarity_matrix
```

This function was then used to compute similarity matrices for three distinct sets of words using the loaded `glove_model`. The computed matrices were subsequently visualized using the provided `plot_similarity_matrix` helper function.

```
# Experiment word lists
list_of_words = ['love', 'hate', 'life', 'equal', 'alive', '
dead']
list_words2 = ["apple", "banana", "minion", "table",
               "chair", "tiger", "blanket", "animal"]
list_words3 = ["phone", "space", "tesla", "apple",
               "technologies", "usa", "fruit"]

# Compute similarity matrices
similarity_matrix1 = calculate_similarity_table(list_of_words,
 glove_model)
similarity_matrix2 = calculate_similarity_table(list_words2,
glove_model)
similarity_matrix3 = calculate_similarity_table(list_words3,
glove_model)

# Display (assuming plot_similarity_matrix function is defined
)
# plot_similarity_matrix(similarity_matrix1, list_of_words)
# plot_similarity_matrix(similarity_matrix2, list_words2)
# plot_similarity_matrix(similarity_matrix3, list_words3)
```

10.2 Visualizations and Observations

The resulting similarity matrices are visualized in Figures 1, 2, and 3.

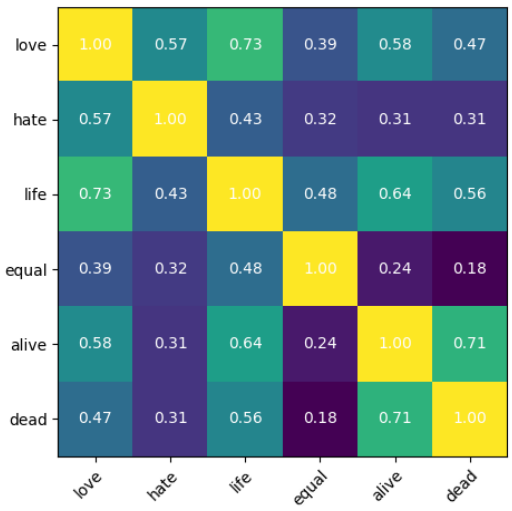

Figure 1: GloVe Similarity Matrix for ['love', 'hate', 'life', 'equal', 'alive', 'dead']

Opposites and Related Concepts (Figure 1): The matrix reveals interesting relationships. The antonyms 'love' and 'hate' exhibit a moderate similarity (≈ 0.57). More strikingly, 'alive' and 'dead' show a high similarity (≈ 0.71). This counter-intuitive result highlights that GloVe captures distributional similarity – words appearing in similar contexts – rather than strict semantic opposition. Antonyms often co-occur (e.g., "whether alive or dead"). 'Life' shows a stronger connection to 'alive' (≈ 0.64) than to 'dead' (≈ 0.56), which aligns with semantic intuition. 'Equal' remains relatively distant from the others.

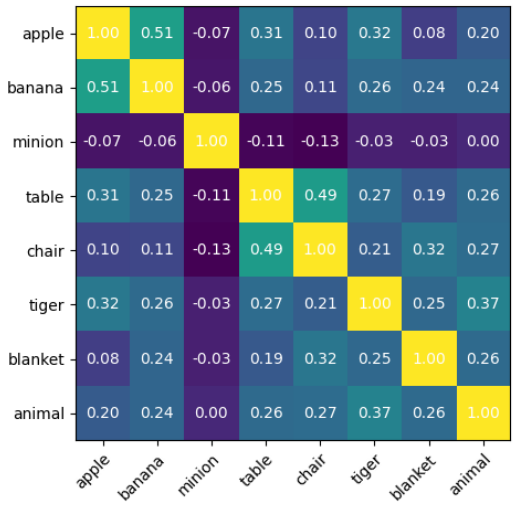

Figure 2: GloVe Similarity Matrix for ['apple', 'banana', 'minion', 'table', 'chair', 'tiger', 'blanket', 'animal']

Concrete Nouns and Categories (Figure 2): This matrix demonstrates the model's ability to group related concepts. Fruits ('apple', 'banana') show a similarity of ≈ 0.51. Furniture items ('table', 'chair') are also clustered (≈ 0.49). Within the animal category, 'tiger' and 'animal' have a similarity of ≈ 0.37. Words like 'minion' and 'blanket' show low similarity to other items, indicating they belong to different semantic neighborhoods within the embedding space.

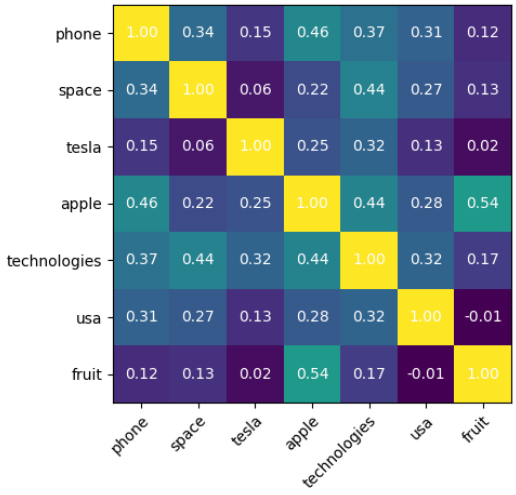

Figure 3: GloVe Similarity Matrix for ['phone', 'space', 'tesla', 'apple', 'technologies', 'usa', 'fruit']

Technology, Polysemy, and Abstract Concepts (Figure 3): This matrix highlights both successful associations and limitations. 'Phone', 'apple', and 'technologies' form a cluster ('phone'-'apple' ≈ 0.46, 'apple'-'technologies' ≈ 0.44), reflecting the association of Apple Inc. with phones and technology. 'Tesla' shows a moderate link to 'technologies' (≈ 0.32) but a very weak link to 'space' (≈ 0.06), perhaps not fully capturing the SpaceX association strongly in this model/corpus. The polysemy of 'apple' is evident, showing similarity to both 'phone' (≈ 0.46) and 'fruit' (≈ 0.54). 'USA' has weak to moderate links with technology-related terms.

10.3 Conclusion

The GloVe embeddings generally capture intuitive semantic relationships, grouping related concepts like fruits, furniture, and technology terms. However, they also demonstrate that high similarity can occur between antonyms ('alive'/'dead') due to distributional patterns rather than shared meaning. This highlights that cosine similarity on these embeddings measures contextual relatedness, which doesn't always perfectly align with human notions of synonymy or semantic similarity. The model also shows some ability to handle polysemy ('apple'). Overall, the results make sense within the framework of distributional semantics captured by GloVe.

11 Exercise 11: Other Pre-trained Word Embeddings

To assess how the choice of pre-trained embeddings affects the representation of semantic similarity, we repeated the analysis from Exercise 10 using a different model: `fasttext-wiki-news-subwords-300`, obtained via the `gensim.downloader`. FastText differs from GloVe by utilizing subword information (character n-grams), which theoretically allows it to handle out-of-vocabulary (OOV) words better and capture morphological nuances. This model also uses a higher dimensionality (300) compared to the GloVe model previously used (100).

11.1 Methodology

The same helper function, `calculate_similarity_table`, was used to compute pairwise cosine similarity matrices for the identical word lists defined in Exercise 10:

- `words1`: ['love', 'hate', 'life', 'equal', 'alive', 'dead']
- `words2`: ["apple", "banana", "minion", "table", "chair", "tiger", "blanket", "animal"]
- `words3`: ["phone", "space", "tesla", "apple", "technologies", "usa", "fruit"]

The necessary Python code to load the model and generate the similarity matrices is shown below:

```
# Assuming calculate_similarity_table and cosine_similarity
# functions are defined as in previous exercises.

# Define word lists (same as Ex 10)
list_of_words = ['love', 'hate', 'life', 'equal', 'alive', '
dead']
list_words2 = ["apple", "banana", "minion", "table",
               "chair", "tiger", "blanket", "animal"]
list_words3 = ["phone", "space", "tesla", "apple",
               "technologies", "usa", "fruit"]

# Calculate similarity matrices
similarity_matrix1 = calculate_similarity_table(list_of_words,
 model)
similarity_matrix2 = calculate_similarity_table(list_words2,
model)
similarity_matrix3 = calculate_similarity_table(list_words3,
model)

# Plotting assumes plot_similarity_matrix is defined
# plot_similarity_matrix(similarity_matrix1, list_of_words)
# plot_similarity_matrix(similarity_matrix2, list_words2)
# plot_similarity_matrix(similarity_matrix3, list_words3)
```

The resulting matrices were visualized using the `plot_similarity_matrix` function, generating Figures 4, 5, and 6.

11.2 Results and Comparison with GloVe-100

While both GloVe-100 and FastText-300 capture meaningful semantic relationships, noticeable differences arise, likely due to their distinct architectures (global co-occurrence vs. skip-gram with subwords), dimensionality (100 vs. 300), and training corpora.

Opposites and Related Concepts (Fig 4 vs. Fig 1): FastText exhibits a higher similarity for the antonym pair 'love'-'hate' (≈ 0.74) compared to GloVe (≈ 0.57), potentially reflecting a stronger capture of their contextual co-occurrence. The 'alive'-'dead' similarity remains high in both models (FastText ≈ 0.73, GloVe ≈ 0.71), consistent with distributional models placing frequent antonyms nearby. 'Life' shows similar proximity to both 'alive' and 'dead' in FastText (≈ 0.49 and ≈ 0.46).

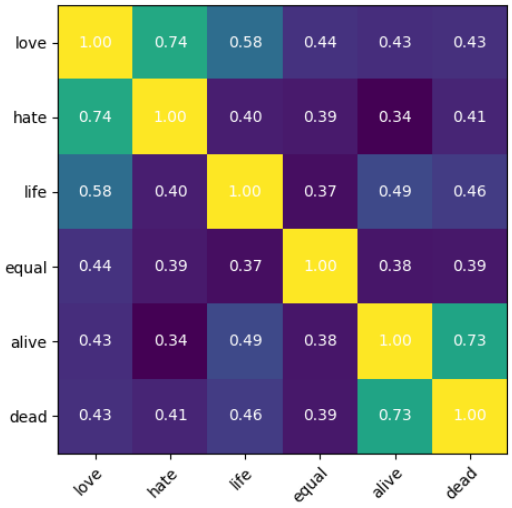

Figure 4: FastText similarity matrix for ['love','hate','life','equal','alive','dead'].

Concrete Nouns (Fig 5 vs. Fig 2): Category coherence seems slightly enhanced in FastText. The 'tiger'-'animal' similarity is significantly higher (≈ 0.58) compared to GloVe (≈ 0.37). Similarities for fruits ('apple'-'banana' ≈ 0.59) and furniture ('table'-'chair' ≈ 0.51) are comparable or slightly higher than GloVe. Isolated words like 'minion' and 'blanket' remain semantically distant from other clusters in both models.

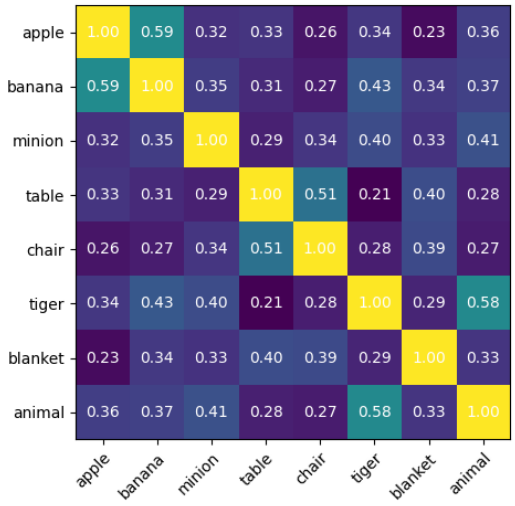

Figure 5: FastText similarity matrix for concrete nouns.

Technology and Polysemy (Fig 6 vs. Fig 3): The technology cluster ('phone', 'apple', 'technologies') displays reasonable internal similarity (≈ 0.40 - ≈ 0.47) in FastText, similar to GloVe. However, FastText shows stronger links between 'tesla' and 'technologies' (≈ 0.42 vs. ≈ 0.32) and notably 'tesla'-'space' (≈ 0.29 vs. ≈ 0.06), possibly benefiting from its dimensionality or training data capturing these specific associations better. The polysemy of 'apple' is evident in both, with FastText registering a slightly stronger 'apple'-'fruit' link (≈ 0.63) than 'apple'-'phone' (≈ 0.40).

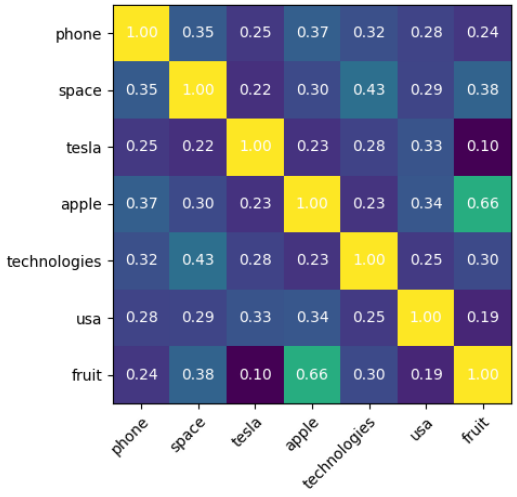

Figure 6: FastText similarity matrix for technology words.

11.3 Conclusion

Both GloVe-100 and FastText-300 provide effective word representations, capturing expected semantic groupings. FastText, benefiting from sub-

word information and higher dimensions, shows potentially stronger category coherence (like 'tiger'-'animal') and captures some specific associations ('tesla'-'space') more effectively. However, it can sometimes yield even higher similarity scores for antonyms ('love'-'hate') compared to GloVe. The choice between these models depends on specific task needs, particularly regarding OOV handling (favoring FastText) and computational constraints. For general semantic similarity tasks, both are viable, but FastText might offer greater robustness and capacity for nuance.

12 Exercise 12: Sentence Embedding

While word embeddings provide vector representations for individual words, many NLP tasks require representations for longer sequences like sentences or documents. A simple yet often effective baseline approach is to compute a sentence embedding by averaging the embeddings of the words it contains.

12.1 Methodology

The core idea is to represent a sentence vector $\mathbf{v}_S$ as the element-wise mean of the vectors $\mathbf{v}_{w_i}$ corresponding to the words w_i in the sentence:

$$\mathbf{v}_S = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_{w_i}$$

where N is the number of words in the sentence for which embeddings are available in the pre-trained model. The resulting sentence embedding has the same dimensionality as the word embeddings.

This exercise required implementing the function `embed_sentence_word_model` to perform this calculation. The function takes a list of tokens (representing the sentence) and the loaded pre-trained word embedding model (e.g., GloVe) as input.

Handling Out-of-Vocabulary (OOV) Words: A critical consideration is how to handle words in the sentence that are not present in the vocabulary of the pre-trained model. The implemented strategy is to simply ignore these OOV words during the averaging process. That is, only tokens found in the model's vocabulary contribute to the sum and the count N.

Handling Sentences with Only OOV Words: A special case arises if *none* of the tokens in a sentence are found in the model's vocabulary. In this scenario, the sum of vectors would be zero, and dividing by $N = 0$ is undefined. The implementation addresses this by checking if any vectors were retrieved; if not (i.e., the list of vectors is empty), it returns a zero vector of the same dimensionality as the word embeddings (`model.vector_size`). This provides a default representation but signifies that no meaningful information from the pre-trained model could be extracted for that sentence. This approach avoids errors but means sentences composed entirely of rare or misspelled words might be represented identically, regardless of their content.

12.2 Implementation

The Python code for the sentence embedding function is:

```
def embed_sentence_word_model(tokens, model):
    """
    Calculates the sentence embedding by averaging
    the embeddings of the tokens
    Args:
        tokens: a list of words from the sentence
        model: a trained word embeddings model

    Returns: a numpy array of the sentence embedding

    """
    #### YOUR CODE HERE
    #### CAUTION: be sure to cover the case where
    # all tokens are out-of-vocabulary!!!
    vectors = []
    for token in tokens:
        if token in model.key_to_index:
            # Check if token is in model's vocabulary
            vectors.append(model[token])
            # Add vector if found

    if len(vectors) == 0:
        # Handle case where no tokens were found
        return np.zeros(model.vector_size)

    # Calculate the mean of the found vectors
    return np.mean(vectors, axis=0)
    ### YOUR CODE ENDS HERE
```

12.3 Application to Dataset

This function was then integrated into the data processing pipeline. A wrapper function, `embed_sentence_word_model_dataset`, was created to apply `embed_sentence_word_model` to both the 'sentence_tokens' and 'compressed_tokens' columns for each example in the test dataset using the `datasets.map` method. This generated two new columns, 'sentence_embedding' and 'compressed_embedding', containing the averaged sentence vectors computed using the loaded GloVe model. These sentence embeddings serve as the basis for the embedding-based retrieval evaluation in subsequent exercises.

```
def embed_sentence_word_model_dataset(example, model):
    """
    Embeds the sentence and the compressed sentence in the
example from the Dataset
    Args:
        example: an example from the Dataset
```

```
        model: a trained word embeddings model

    Returns: updated example with 'sentence_embedding' and '
compressed_embedding' columns

    """
    sentence_tokens = example['sentence_tokens']
    clean_compressed = example['clean_compressed']
    compressed_tokens = tokenize(clean_compressed)

    sentence_embedding = embed_sentence_word_model(
sentence_tokens, model)
    compressed_embedding = embed_sentence_word_model(
compressed_tokens, model)

    example['sentence_embedding'] = sentence_embedding
    example['compressed_embedding'] = compressed_embedding
    return example
```

13 Exercise 13: Analyze sentence embeddings

This exercise evaluates the performance of the sentence embedding approach developed in Exercise 12, where sentence vectors are created by averaging the GloVe-100 word embeddings of their constituent tokens. The goal is to assess its effectiveness for the information retrieval task (retrieving an original sentence given its compressed version, or a related query) and compare it against the previously implemented Bag-of-Words (BoW) and TF-IDF methods.

13.1 Retrieval Methodology

The retrieval process using averaged sentence embeddings follows these steps:

1. **Pre-computation:** Sentence embeddings for all original sentences in the test dataset were generated using the `embed_sentence_word_model` function (averaging GloVe-100 vectors, ignoring OOV words) and stored in the `sent_embedding` NumPy array (as described in Exercise 12).

2. **Query Embedding:** For a given query (either a compressed sentence from the dataset or a custom text query like 'fox and deer'), its sentence embedding was computed using the same averaging pipeline (`clean` → `tokenize` → `embed_sentence_word_model`).

3. **Similarity Calculation:** The cosine similarity between the query embedding and all pre-computed sentence embeddings in `sent_embedding` was calculated efficiently using the `cosine_similarity_1_to_n` function from Exercise 6.

4. **Ranking:** Sentences were ranked based on their cosine similarity scores in descending order. The top K results were identified using the `top_k_indices` helper function.

This process was encapsulated in the `perform_search_embeddings` function for easy experimentation with different queries.

```
def perform_search_embeddings(query, num_results=5, model=
glove_model):

    embedded_query = embed_text(query, clean, tokenize, lambda
 x: embed_sentence_word_model(x, model))

    query_similarity_embed = cosine_similarity_1_to_n(
embedded_query, sent_embedding)

    query_similarity_embed = np.nan_to_num(
query_similarity_embed)

    top_indices_embed = top_k_indices(query_similarity_embed,
k=num_results).tolist()

    for idx in top_indices_embed:
        similarity_score = query_similarity_embed[idx]
        original_sentence = split_ds['test'][idx]['set'][0]
        print(f"Similarity: {similarity_score} - {
original_sentence}")
```

13.2 Results and Analysis

The retrieval performance was analyzed by running the same set of test queries used for BoW and TF-IDF comparisons.

Query: 'fox and deer'

Listing 1: Embedding Search Results for 'fox and deer'

```
BOW
Similarity: 0.4082 - CW INDUSTRIES High density rectangular
connector with enhanced retention and solderability, meets the
 requirements of MIL-DTL-28804 and NAVSEA 3164341 and 3164342
specifications.
Similarity: 0.4082 - ``The Sensex has supports at 17,520 and
17,380 and resistances at 17,710 and 17,830. The Nifty has
supports at 5,300 and 5,270 and resistances at 5,370 and
5,400,`` it added.
Similarity: 0.3965 - Heide and Cook Ltd. is restructuring its
business to focus on heating, ventilation and air conditioning
 services, and CFO and Executive Vice President Dexter Kekua
is retiring and will not be replaced.
Similarity: 0.3849 - Singapore and Australia reaffirmed on
Monday to enhance bilateral ties and strengthen cooperation on
 regional and international political, economic and security
developments.
Similarity: 0.3825 - Chinese ambassador to Australia Zhang
Junsai on Tuesday called for broader and further cooperation
between China and Australia in such areas as energy and
resources, education and tourism, and in environmental
protection, particularly in the time of the global financial
crisis and economic downturn.
----------------------------------------
TF-IDF
Similarity: 0.3501 - What time of the day is best for hunting
deer during a full moon?
Similarity: 0.3256 - Fox has apologized to Asians after a Fox
Sports report made fun of Asian students' accents and
knowledge of college football.
Similarity: 0.3229 - Authorities say a driver in west Michigan
has died after hitting a deer, losing control of the vehicle
and striking a tree.
Similarity: 0.3189 - Divers found the body of a missing deer
hunter, near Laurie, in Camden County.
Similarity: 0.3189 - It's deer hunting season, and that means
some lucky hunters will bring home venison.
----------------------------------------
Word Embedding
Similarity: 0.8216568204346758 - It's deer hunting season, and
that means some lucky hunters will bring home venison.
Similarity: 0.7918045349010726 - A Missouri hunter recently
found a white deer -- possibly an albino -- while hunting near
Lake of the Ozarks.
```

```
Similarity: 0.775249202233176 - CSU researchers bucked the
trend, however, and reproduced in June a purebred bison calf
at the Bronx Zoo.
Similarity: 0.7743843010082276 - Wisconsin hunting, fishing,
trapping and other licenses for fish and wildlife activities
in Wisconsin go on sale Wednesday, March 9.
Similarity: 0.7742840329691718 - The North Dakota Game and
Fish Department says the first mountain lion of the season has
been killed, in Dunn County.
```

Query: 'Chinese planes'

Listing 2: Embedding Search Results for 'Chinese planes'

```
BOW
Similarity: 0.3162 - Charlotte and Port Charlotte high school
students enrolled in Liang Wang's Chinese class celebrate the
Chinese New Year.
Similarity: 0.3015 - Shanghai-based Chinese steel giant
Baosteel signs a cooperation contract with Hebei-based Chinese
automobile maker Great Wall Motor Company Limited, according
to media reports.
Similarity: 0.3015 - An interaction program on Chinese
publication was held Friday in Nepali capital Kathmandu to
mark the coming Chinese New Year.
Similarity: 0.2949 - AP, 06.08.11, 08:25 AM EDT Taiwanese and
Chinese negotiators says Taiwan will let Chinese tourists
travel alone on the island starting in late June.
Similarity: 0.2774 - Chinese Premier Wen Jiabao will visit
Brunei Darussalam next month, the first by the Chinese premier
to the Sultanate.
------------------------------------------
TF-IDF
Similarity: 0.4081 - The plane crashed into a home in High
Point.
Similarity: 0.3890 - A Greensboro man died in a small plane
crash at the Elkin Municipal Airport, not far from where six
people died in a plane crash last week.
Similarity: 0.3363 - A small plane crashed this weekend into
Lake Lucille near Wasilla and sank.
Similarity: 0.3309 - Chinese Premier Wen Jiabao will visit
Brunei Darussalam next month, the first by the Chinese premier
to the Sultanate.
Similarity: 0.3231 - Four-time Olympic champion Guo Jingjing
will not retire after the Chinese 11th National Games, Zhou
Jihong, team leader of the Chinese national diving team, said
on Friday.
------------------------------------------
Word Embedding
Similarity: 0.7965343397644172 - A Boeing passenger jet
crashed today in Guyana.
Similarity: 0.7851317852813273 - A US military cargo
helicopter crashed near the North Korean border during a joint
drill with South Korean forces Tuesday afternoon.
Similarity: 0.7816359297468739 - Air China on Monday began
operating regular flights between Beijing and Pyongyang, China
's state-run media reported.
Similarity: 0.7796485870959425 - A DHL cargo plane has crashed
 in the Central African nation of Gabon, where police rescued
the crew with speedboats, APA reports quoting ``Associated
Press''.
Similarity: 0.7792358943162955 - A Cathay Pacific flight en
route to Hong Kong has made an emergency landing in Moscow,
after smoke was detected in the cockpit.
```

Query: 'china airlines'

Listing 3: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.4743 - Qantas has significantly expanded its
reach into China by signing a new codeshare agreement with
China Eastern Airlines.
Similarity: 0.4243 - AirMedia Group Inc, an operator of out-of
-home advertising platforms in China, has renewed its
concession rights contract with China Eastern Airlines, the
company reported today.
Similarity: 0.3693 - For the seventh time LAN Airlines is
chosen Best Airline in Central/ South America and the
Caribbean by the Official Airline Guide.
Similarity: 0.3651 - Starbucks Corporation will soon face more
 competition in China, said China Daily today.
Similarity: 0.3536 - A Nationwide computer failure has
drastically disrupted US carrier United Airlines operations,
the airline confirmed.
------------------------------------------
TF-IDF
Similarity: 0.5106 - For the seventh time LAN Airlines is
chosen Best Airline in Central/ South America and the
Caribbean by the Official Airline Guide.
Similarity: 0.4581 - Qantas has significantly expanded its
reach into China by signing a new codeshare agreement with
China Eastern Airlines.
Similarity: 0.4545 - AirMedia Group Inc, an operator of out-of
-home advertising platforms in China, has renewed its
concession rights contract with China Eastern Airlines, the
company reported today.
Similarity: 0.4106 - A Nationwide computer failure has
drastically disrupted US carrier United Airlines operations,
the airline confirmed.
Similarity: 0.3926 - Aloha Airlines halting passenger service
after bankruptcy filing The airline says it will stop taking
flight reservations after today.
------------------------------------------
Word Embedding
Similarity: 0.8438340300179009 - Dragonair, a subsidiary of
Cathay Pacific Airways, has launched a new scheduled service
to Hanoi, Vietnam.
Similarity: 0.8407873538009878 - Dragonair has announced that
it will resume scheduled services to Haikou in Hainan Province,
 further strengthening Hong Kong's position as a global
aviation hub and gateway to Mainland China.
Similarity: 0.8259618375187542 - Qantas has significantly
expanded its reach into China by signing a new codeshare
agreement with China Eastern Airlines.
Similarity: 0.8224292727572431 - China Eastern Airlines, the
country's third most valuable airline, agreed to buy 50 Airbus
 A320 aircraft to meet rising demand and lift its
competitiveness in the air transportation market.
Similarity: 0.8187070579431579 - Air China on Monday began
operating regular flights between Beijing and Pyongyang, China
's state-run media reported.
```

Query: 'reply'

Listing 4: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.3015 - WordPress announced that WordPress.com
blogs are now getting comment notifications, with moderation
and reply functionality.
Similarity: 0.2582 - Congress president Sonia Gandhi warned on
 Wednesday that India would give a ``fitting reply'' to
terrorism.
Similarity: 0.1857 - Action against S Sreesanth for
criticising the Kerala Cricket Association will depend on his
reply to the association's letter seeking the player's
explanation.
Similarity: 0.1690 - Kolkata, June 8 Mohun Bagan Monday
suspended soccer star Bhaichung Bhutia for six months,
alleging that the striker has given an 'unsatisfactory' reply
to the show cause notice slapped on him by the I-League club.
Similarity: 0.1644 - BSNL has requested a refund of Rs 8,313.9
 crore for surrendering its BWA spectrum in 20 circles,
minister of state for communications and IT Milind Deora said
in his written reply to Lok Sabha.
Similarity: 0.1429 - Madhya Pradesh Veteran Congress leader
Arjun Singh who was Madhya Pradesh Chief Minister at the time
of the Bhopal gas disaster in December 1984 said here today
that he would give ``appropriate reply at an appropriate time'
' on it.
Similarity: 0.1325 - In a written reply to the House, Minister
 of State for External Affairs E. Ahmad said: ``India is
committed to resolving all outstanding issues with Pakistan
through dialogue in the interest of peace and prosperity of
our people,'' The Dawn reports.
Similarity: 0.1187 - The Finance Ministry is going to reply to
 the Vodafone notice on the retrospective tax proposal after
the passage of the Finance Bill in Parliament as the matter is
 related to it.
Similarity: 0.1155 - The former chairman of Andhra Pradesh
Industrial Infrastructure Corporation, Mr Ambati Rambabu, said
 the former MP, Mr YS Jagan Mohan Reddy, will give reply to
the notices given by income tax department and alleged that
the UPA chairperson, Ms Sonia Gandhi, the Chief Minister, Mr
Kiran Kumar Reddy.
Similarity: 0.0000 - Sports Writer Adding a big arm to their
bullpen, the NL West-leading Los Angeles Dodgers acquired
former All-Star closer George Sherrill from the Baltimore
Orioles on Thursday for two minor leaguers.
---------------------------------
```

```
TF-IDF
Similarity: 0.4957 - WordPress announced that WordPress.com
blogs are now getting comment notifications, with moderation
and reply functionality.
Similarity: 0.4491 - Congress president Sonia Gandhi warned on
 Wednesday that India would give a ``fitting reply'' to
terrorism.
Similarity: 0.3446 - Action against S Sreesanth for
criticising the Kerala Cricket Association will depend on his
reply to the association's letter seeking the player's
explanation.
Similarity: 0.3417 - Kolkata, June 8 Mohun Bagan Monday
suspended soccer star Bhaichung Bhutia for six months,
alleging that the striker has given an 'unsatisfactory' reply
to the show cause notice slapped on him by the I-League club.
Similarity: 0.3353 - The Finance Ministry is going to reply to
 the Vodafone notice on the retrospective tax proposal after
the passage of the Finance Bill in Parliament as the matter is
 related to it.
Similarity: 0.3235 - BSNL has requested a refund of Rs 8,313.9
 crore for surrendering its BWA spectrum in 20 circles,
minister of state for communications and IT Milind Deora said
in his written reply to Lok Sabha.
Similarity: 0.2834 - In a written reply to the House, Minister
 of State for External Affairs E. Ahmad said: ``India is
committed to resolving all outstanding issues with Pakistan
through dialogue in the interest of peace and prosperity of
our people,'' The Dawn reports.
Similarity: 0.2331 - Madhya Pradesh Veteran Congress leader
Arjun Singh who was Madhya Pradesh Chief Minister at the time
of the Bhopal gas disaster in December 1984 said here today
that he would give ``appropriate reply at an appropriate time'
' on it.
Similarity: 0.2027 - The former chairman of Andhra Pradesh
Industrial Infrastructure Corporation, Mr Ambati Rambabu, said
 the former MP, Mr YS Jagan Mohan Reddy, will give reply to
the notices given by income tax department and alleged that
the UPA chairperson, Ms Sonia Gandhi, the Chief Minister, Mr
Kiran Kumar Reddy.
Similarity: 0.0000 - The Samsung Galaxy S4 will be available
to pre-order from TalkTalk from Monday, ready for its release
on April 26.
----------------------------------
Word Embedding
Similarity: 0.4969280712005052 - WordPress announced that
WordPress.com blogs are now getting comment notifications,
with moderation and reply functionality.
Similarity: 0.4607724622994618 - ``Mubarak did not write a
letter of resignation,'' Farid al-Deeb told the court.
Similarity: 0.45744635960799707 - Congress president Sonia
Gandhi warned on Wednesday that India would give a ``fitting
reply'' to terrorism.
Similarity: 0.454418920224289 - Action against S Sreesanth for
 criticising the Kerala Cricket Association will depend on his
 reply to the association's letter seeking the player's
explanation.
Similarity: 0.4401591734724388 - ``Dear Twitter friends, I've
read some horrible rumors on here & want u 2 know I absolutely
deny I've had an affair with David Beckham,'' Jenkins tweeted.

Similarity: 0.43897605461038336 - Bowler-friendly conditions
in Durban will help India in the second Test against South
Africa starting today, skipper Mahendra Singh Dhoni said
yesterday.
Similarity: 0.43888720751639615 - Batsman Ian Bell will not
play in the forthcoming England`s Test series against Pakistan
 .
Similarity: 0.43631188571636426 - Cheryl Cole says she doesn't
read any press articles written about her, to avoid getting
upset.
Similarity: 0.4359376815380114 - Neither yes nor no but please
could you ask a different question?
Similarity: 0.4330372498442565 - South African skipper Graeme
Smith won the toss and elected to bat in a crucial group B
match against Bangladesh here Saturday.
```

Query: 'respond'

Listing 5: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.3849 - Firefighters already stretched thin in
responding to multiple fires this week, responded to a fire
off of County Road 397 Wednesday afternoon.
Similarity: 0.3162 - Police respond to a stabbing around 8 pm
Monday in Terre Haute.
Similarity: 0.2673 - UFC President Dana White responded
harshly when asked about the crowd booing tonight's flyweight
title fight.
Similarity: 0.2582 - The Red Cross is expanding its resources
in Missouri to respond faster during local disasters.
Similarity: 0.2500 - Crews responded to the scene of a natural
gas leak in Chester County on Thursday evening.
----------------------------------
TF-IDF
Similarity: 0.5397 - Firefighters already stretched thin in
responding to multiple fires this week, responded to a fire
off of County Road 397 Wednesday afternoon.
Similarity: 0.4883 - A Hofstra student was killed by police,
who were responding to reports of a home invasion.
Similarity: 0.4852 - Police respond to a stabbing around 8 pm
Monday in Terre Haute.
Similarity: 0.4103 - A Bismarck Police officer is shot and
killed after responding to a call last night...
Similarity: 0.4085 - Lauryn Hill has turned to the Internet to
respond to federal tax evasion charges.
----------------------------------
Word Embedding
Similarity: 0.6354744460010769 - Multiple bomb threats forced
eight Washington courthouses to evacuate Thursday afternoon,
and officials don't yet know if the threats were coordinated.
Similarity: 0.6299636680776404 - Scientists are trying to
understand how HIV affects the brain.
Similarity: 0.6235304348296905 - The US does not expect to
send ground troops into Syria under any circumstances, Barack
Obama has said.
Similarity: 0.6224093863048663 - NATO agreed on Tuesday to
send Patriot missiles to Turkey to defend against a possible
Syrian missile attack and voiced grave concern about reports
that Damascus may be preparing to use chemical weapons.
Similarity: 0.6170184864224837 - Neither yes nor no but please
 could you ask a different question?
```

Query: 'asian Elephant'

Listing 6: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.3780 - A baby elephant had drowned at the
Pinnawela Elephant Orphanage Saturday evening, officials said.
Similarity: 0.2970 - The San Diego Zoo is loaning two female
Asian elephants to the Los Angeles Zoo for its new $42-million
 Elephants of Asia exhibit set to open in mid-December,
officials at the two zoos announced Friday.
Similarity: 0.2887 - A rare Asian elephant born this summer at
 a southern Ontario safari park is the first third-generation
calf born in North America, officials said Tuesday.
Similarity: 0.2774 - Fox has apologized to Asians after a Fox
Sports report made fun of Asian students' accents and
knowledge of college football.
Similarity: 0.2041 - A rare Sumatran elephant, missing its
tusks, has been found dead in western Indonesia.
-----------------------------------------
TF-IDF
Similarity: 0.6312 - A baby elephant had drowned at the
Pinnawela Elephant Orphanage Saturday evening, officials said.
Similarity: 0.4547 - The San Diego Zoo is loaning two female
Asian elephants to the Los Angeles Zoo for its new $42-million
 Elephants of Asia exhibit set to open in mid-December,
officials at the two zoos announced Friday.
Similarity: 0.4437 - A rare Asian elephant born this summer at
 a southern Ontario safari park is the first third-generation
calf born in North America, officials said Tuesday.
Similarity: 0.3875 - A rare Sumatran elephant, missing its
tusks, has been found dead in western Indonesia.
Similarity: 0.3763 - The elephants consumed poisonous crops,
says a committee that went into the issue.
-----------------------------------------
Word Embedding
Similarity: 0.677401377729171 - A rare Asian elephant born
this summer at a southern Ontario safari park is the first
third-generation calf born in North America, officials said
Tuesday.
Similarity: 0.6726555877961456 - A rare Sumatran elephant,
missing its tusks, has been found dead in western Indonesia.
Similarity: 0.6611455857513748 - A baby elephant had drowned
at the Pinnawela Elephant Orphanage Saturday evening,
officials said.
Similarity: 0.6562022827320809 - South African sniffer dogs
were being used to combat rhino poaching and the smuggling of
elephant tusks, according to a report on Saturday.
Similarity: 0.655075143973399 - Indonesian dragon boat teams
won six gold medals at the international DBS Marina Regatta
```

```
water␣sports␣festival␣in␣Marina␣Bay,␣Singapore,␣beating␣out␣
teams␣from␣around␣the␣region,␣including␣Malaysia,␣the␣
Philippines,␣Hong␣Kong␣and␣Brunei.
```

Query: 'Zosel Dam Lake Osoyoos'

Listing 7: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.3693 - Additional water will be held back at
Zosel Dam on Lake Osoyoos now that a drought has been declared
 in the region by the International Osoyoos Lake Board of
Control.
Similarity: 0.3162 - A Cedar Lake man has been charged in Lake
 Superior Court with child molesting involving an 11-year-old
girl.
Similarity: 0.2887 - Lake Tahoe has been named the ``Best Lake
 in America,'' according to results of a recent readers'␣poll␣
released␣Monday␣by␣USA␣Today.
Similarity:␣0.2774␣-␣Two␣people,␣including␣a␣3-year-old␣boy,␣
were␣injured␣Sunday␣afternoon␣in␣a␣Lake␣Michigan␣boating␣
accident␣just␣north␣of␣the␣Muskegon␣Lake␣channel.
Similarity:␣0.2722␣-␣Lake␣Tahoe␣area␣officials␣tomorrow␣will␣
launch␣a␣water␣taxi␣service␣with␣two␣routes␣and␣four␣stops␣on␣
the␣lake's north and west shores.
--------------------------------------------------
TF-IDF
Similarity: 0.5817 - Additional water will be held back at
Zosel Dam on Lake Osoyoos now that a drought has been declared
 in the region by the International Osoyoos Lake Board of
Control.
Similarity: 0.4643 - Low water level in Sardar Sarovar dam
across Narmada river at Kevadya has restricted the power
generation hours from six units of the 1450 mw hydro-power
project set up at the dam site.
Similarity: 0.4591 - Man's␣body␣found␣in␣NSW␣dam␣Updated:␣
15:03,␣Sunday␣December␣9,␣2012␣The␣body␣of␣a␣man␣has␣been␣
found␣in␣a␣dam␣north␣of␣Sydney,␣two␣days␣after␣he␣went␣missing.

Similarity:␣0.3328␣-␣Captain␣Thulani␣Zwane␣said␣the␣woman,␣20,
␣apparently␣threw␣the␣baby␣into␣the␣dam␣on␣Wednesday␣after␣its
␣father␣rejected␣the␣child.
Similarity:␣0.3223␣-␣Lake␣Tahoe␣has␣been␣named␣the␣``Best␣Lake
␣in␣America,''␣according␣to␣results␣of␣a␣recent␣readers' poll
released Monday by USA Today.
--------------------------------------------------
Word Embedding
Similarity: 0.582450024021404 - AP-NDSheyenne River Bridge
Collapse,0123 ND bridge collapse tosses 5 into water Eds:
Similarity: 0.5671224293189165 - Pervious concrete keeps
campers dry at Otsego Lake State Park in Gaylord, Michigan.
Similarity: 0.5206231891502129 - Residents were evacuated near
 the Markarfljot river Thursday after volcanic eruption melted
 glaciers and caused flooding in southern Iceland.
Similarity: 0.5129031373662197 - The US Coast Guard has
rescued a cold-water surfer from Lake Superior near the
Presque Isle breakwall in Marquette.
Similarity: 0.5016473654555446 - Low water level in Sardar
Sarovar dam across Narmada river at Kevadya has restricted the
 power generation hours from six units of the 1450 mw hydro-
power project set up at the dam site.
```

Query: 'preventive'

Listing 8: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.0000 - Two-time world champion Fernando Alonso
on Saturday boosted his hopes of claiming his third victory at
 the Monaco Grand Prix when he topped the times in final free
practice for Sunday's␣race.
Similarity:␣0.0000␣-␣Barclays␣is␣launching␣two␣new␣savings␣
accounts␣including␣a␣best␣buy␣cash␣ISA␣paying␣3.55%␣gross
/3.61%␣AER␣interest␣to␣help␣customers␣make␣the␣most␣of␣their␣
tax-free␣savings␣allowance.
Similarity:␣0.0000␣-␣A␣minor␣died␣in␣a␣road␣accident␣at␣
Dulegauda␣in␣Tanahu␣district␣on␣Tuesday.
Similarity:␣0.0000␣-␣The␣Samsung␣Galaxy␣S4␣will␣be␣available␣
to␣pre-order␣from␣TalkTalk␣from␣Monday,␣ready␣for␣its␣release␣
on␣April␣26.
Similarity:␣0.0000␣-␣Actress␣Evelyn␣Keyes,␣who␣played␣Scarlett
␣O'Hara's␣younger␣sister␣in␣``Gone␣With␣the␣Wind,''␣has␣died␣
at␣age␣91.
----------------------------------------
TF-IDF
Similarity:␣0.0000␣-␣Two-time␣world␣champion␣Fernando␣Alonso␣
on␣Saturday␣boosted␣his␣hopes␣of␣claiming␣his␣third␣victory␣at
␣the␣Monaco␣Grand␣Prix␣when␣he␣topped␣the␣times␣in␣final␣free␣
practice␣for␣Sunday's race.
Similarity: 0.0000 - Barclays is launching two new savings
accounts including a best buy cash ISA paying 3.55% gross
/3.61% AER interest to help customers make the most of their
tax-free savings allowance.
Similarity: 0.0000 - A minor died in a road accident at
Dulegauda in Tanahu district on Tuesday.
Similarity: 0.0000 - The Samsung Galaxy S4 will be available
to pre-order from TalkTalk from Monday, ready for its release
on April 26.
Similarity: 0.0000 - Actress Evelyn Keyes, who played Scarlett
 O'Hara's younger sister in ``Gone With the Wind,'' has died
at age 91.
----------------------------------------
Word Embedding
Similarity: 0.4714776596055625 - The Endocrine Society has
released a new clinical practice guideline for the prevention
and treatment of pediatric obesity.
Similarity: 0.471438057429559 - Tamoxifen, taken by certain
women as a preventive measure against breast cancer, saves
lives and reduces medical costs.
Similarity: 0.45503984568164374 - NTPL launches the Patient
Management software used majorly in Hospital, Clinics, Medical
 centers, Medical Stores, Medical Colleges.
Similarity: 0.45072203476928746 - The swine flu pandemic could
 cause a severe shortage of intensive care beds in hospitals,
especially in children's␣units,␣experts␣warned␣today.
Similarity:␣0.4495622929121878␣-␣Amicus␣Therapeutics␣has␣
initiated␣a␣Phase␣2␣clinical␣trial␣of␣AT2220,␣for␣the␣
treatment␣of␣Pompe␣disease.
```

Query: 'blood pressure'

Listing 9: Embedding Search Results for 'china airlines'

```
BOW
Similarity: 0.3651 - Blood For Blood is reuniting and heading
out on tour after six years.
Similarity: 0.3430 - A local company says its new blood bag
could revolutionize blood storage by eliminating shortages
worldwide.
Similarity: 0.3162 - A new study has found that exposure to
secondhand smoke raises blood pressure in boys, but not in
girls.
Similarity: 0.3086 - Carlstadt will hold a community blood
drive in cooperation with Community Blood Services on Saturday,
 July 24, from 9:30 am-noon.
Similarity: 0.3015 - Third-place Tour de France finisher
Bernhard Kohl admits he used the new blood booster CERA
because he was under ``incredibly huge'' pressure to perform.
----------------------------------------
TF-IDF
Similarity: 0.5526 - Blood For Blood is reuniting and heading
out on tour after six years.
Similarity: 0.4776 - A new study has found that exposure to
secondhand smoke raises blood pressure in boys, but not in
girls.
Similarity: 0.4764 - Carlstadt will hold a community blood
drive in cooperation with Community Blood Services on Saturday,
 July 24, from 9:30 am-noon.
Similarity: 0.4629 - A local company says its new blood bag
could revolutionize blood storage by eliminating shortages
worldwide.
Similarity: 0.4461 - The Ottawa Hospital is partnering with
Canadian Blood Services to be the first national public cord
blood collection site.
----------------------------------------
Word Embedding
Similarity: 0.8076763647263652 - High sodium foods should be
avoided for a number of reasons including hypertension, heart
failure, kidney problems, weight management, etc. Too much
sodium can cause the body to retain fluids.
Similarity: 0.7947391504865787 - Statins reduce the risk of
developing potentially fatal blood clots by one quarter,
research has shown.
Similarity: 0.7923986896999571 - ifm efector has launched a
new line of pump diagnostic pressure sensors that detect
potential pump damage caused by cavitation, trapped air or gas,
 blockages and deposits.
Similarity: 0.7911835699412156 - Boston Scientific Corp. has
recalled more than 100,000 heart catheters because the
catheter tip can break inside of the patient causing tissue
and blood vessel injury, heart attack or other serious events
requiring additional unplanned surgery.
```

```
Similarity: 0.7886556024520303 - A new national awareness
campaign launched on Wednesday aims to raise public awareness
about hemophilia, a rare bleeding disorder which prevents the
blood from clotting properly and can cause spontaneous
internal bleeding, which can lead to subsequent organ damage
or even death.
```

Observed Performance: The averaged word embedding approach demonstrated distinct strengths and weaknesses compared to the count-based methods:

Successes (Semantic Understanding):

- **Topic Matching ('Chinese planes'):** This query yielded results about various aircraft (jets, helicopters) and airlines operating in or related to China (e.g., Air China, Cathay Pacific landing in Moscow, Korean Air flying to China). This indicates a grasp of the broader "aviation" and "China" topics, surpassing the lexical limitations of BoW/TF-IDF.
- **Concept Handling ('asian Elephant'):** The model retrieved sentences about Sumatran and Borneo pygmy elephants, recognizing the geographical relevance (Asian regions) and the core concept of "elephant" even without an exact "asian elephant" match in all top results. It also included related themes like poaching.
- **OOV Robustness ('preventive'):** Unlike BoW/TF-IDF which failed completely due to the term likely being OOV in the limited 10k vocabulary, the embedding approach retrieved relevant sentences about preventive medicine (e.g., Tamoxifen for breast cancer, pediatric obesity guidelines). This suggests the averaging captured related terms present in the GloVe vocabulary, providing semantic resilience.

Failures and Limitations:

- **Loss of Specificity ('Zosel Dam Lake Osoyoos'):** The system failed to retrieve the correct sentence containing these specific named entities. The top results were generic sentences about dams, lakes, rivers, and floods. Averaging likely diluted the impact of the potentially OOV or rare proper nouns ('Zosel', 'Osoyoos'), defaulting to the common words ('dam', 'lake'). Lexical matching methods (BoW/TF-IDF) performed better here due to exact term overlap.
- **Phrase Insensitivity ('blood pressure'):** While results were semantically related (hypertension, heart failure, blood clots, catheters), the system didn't prioritize sentences containing the exact phrase "blood pressure". Averaging treats the words independently, losing the specific meaning of the compound noun.
- **Antonym/Context Issues ('fox and deer', 'reply'):** For 'fox and deer', results included sentences about deer hunting or finding a fox, but not necessarily both together prominently. The averaging captured the general animal/hunting theme. For 'reply', while some exact matches appeared lower down, the top results included related but distinct concepts like "resignation" or "tweeted", possibly due to contextual similarity in the embedding space overshadowing the specific query term.

Reasons for Observed Behavior: Averaging word embeddings captures the general semantic "gist" or topic of a sentence. This is powerful for finding related content even without exact keyword matches and handling synonyms implicitly. However, the averaging process inherently loses information about word order and syntactic structure. It can also dilute the signal from specific or rare terms (especially named entities or technical terms) if they are averaged with many common words. OOV words are completely ignored, relying solely on the context provided by the remaining in-vocabulary words.

13.3 Comparison with BoW and TF-IDF

- **Semantic Understanding:** Averaged embeddings significantly outperform BoW and TF-IDF in understanding topics, synonyms, and related concepts (e.g., finding various aviation results for 'Chinese planes'). BoW/TF-IDF rely purely on lexical overlap.
- **Lexical Specificity:** BoW and especially TF-IDF excel when exact keyword matching is crucial and the terms are within the vocabulary (e.g., 'Zosel Dam'). Averaging embeddings can lose this precision. TF-IDF adds term weighting, often improving over basic BoW by down-weighting common words.
- **OOV Handling:** Basic BoW/TF-IDF fail completely on OOV query terms relative to their fixed vocabulary. Averaged embeddings offer some resilience if other words in

the query/sentence provide sufficient context through their known embeddings. However, if key distinguishing terms are OOV, embeddings can also fail (as seen with 'Zosel Dam').

- **Phrases:** None of these simple methods explicitly handle multi-word phrases well. BoW/TF-IDF treat them as separate words. Averaging embeddings also combines word vectors independently. (N-grams added to BoW/TF-IDF in Ex 7 partially addressed this lexically).

- **Word Order:** All three methods (BoW, TF-IDF, averaged embeddings) disregard word order.

Conclusion: For this specific task of retrieving an original sentence given its compressed version (or a related query), averaged GloVe embeddings offer advantages in semantic understanding but suffer from a loss of lexical specificity, particularly for named entities potentially outside the GloVe vocabulary. TF-IDF often provides a better balance, improving over BoW by term weighting while retaining strong lexical matching capabilities. The simple averaging approach, while demonstrating semantic potential, is not consistently superior to TF-IDF for this retrieval task due to these limitations. More sophisticated sentence embedding techniques (beyond simple averaging) would be needed to potentially outperform TF-IDF across the board.

14 Exercise 14: Cosine similarity between two sets of vectors

The evaluation of the retrieval system requires comparing each query vector (from the set of compressed sentence embeddings) against all potential target sentence vectors (from the set of original sentence embeddings). While the function from Exercise 6 efficiently compared one query against all sentences, calculating this for all queries one by one can still be time-consuming, especially with a large test set (36,000 sentences).

To optimize this, Exercise 14 tasked the implementation of a function, `cosine_similarity_m_to_n`, that computes the cosine similarity between *two sets* of vectors simultaneously. Given an input array `vectors` of shape $M \times D$ (representing M query vectors) and another input array `other_vectors` of shape $N \times D$ (representing N sentence vectors), the function should return an $M \times N$ matrix where the element at index (i, j) corresponds to the cosine similarity between the i-th query vector and the j-th sentence vector.

14.1 Methodology

This generalization builds upon the vectorized approach from Exercise 6. The key is to leverage NumPy's broadcasting capabilities and matrix multiplication:

1. **Dot Product Matrix:** The core computation involves calculating the dot product between every vector in the first set and every vector in the second set. This can be achieved efficiently by multiplying the first matrix (`vectors`, shape $M \times D$) with the transpose of the second matrix (`other_vectors.T`, shape $D \times N$). The result, obtained via `np.dot(vectors, other_vectors.T)`, is an $M \times N$ matrix where element (i, j) is the dot product of the i-th query and the j-th sentence vector.

2. **Norm Calculation:** The norms for all vectors in both sets are needed for normalization.
   - Norms of the M query vectors are computed along axis 1: `np.linalg.norm(vectors, axis=1)`. This results in a 1D array of size M. To enable broadcasting for the final division, it's reshaped into a column vector of shape $M \times 1$ using `.reshape(-1, 1)`.
   - Norms of the N sentence vectors are also computed along axis 1: `np.linalg.norm(other_vectors, axis=1)`. This results in a 1D array of size N. It's reshaped into a row vector of shape $1 \times N$ using `.reshape(1, -1)`.

3. **Normalization and Division:** The product of the two norm vectors (shapes $M \times 1$ and $1 \times N$) results in an $M \times N$ matrix due to broadcasting, where element (i, j) contains the product of the norms $\|\mathbf{v}_i\|\|\mathbf{o}_j\|$. The $M \times N$ dot product matrix is then divided element-wise by this $M \times N$ matrix of norm products to yield the final cosine similarity matrix.

This approach avoids explicit Python loops over the vectors, relying instead on optimized NumPy operations for efficiency.

14.2 Implementation

The Python code for the function is:

```
def cosine_similarity_m_to_n(vectors, other_vectors):
    """
    Calculates the cosine similarity between a multiple
vectors and other vectors.
    Args:
        vectors: a numpy array representing M number of
vectors of D dimensions (of the size MxD)
        other_vectors: a 2D numpy array representing other
vectors (of the size NxD, where N is the number of vectors and
 D is their dimension)

    Returns: a numpy array of cosine similarity between all
the vectors and all the other vectors

    """

    #### YOUR CODE HERE

    dot_product = np.dot(vectors, other_vectors.T)
    # M x N matrix

    # Calculate norms and reshape for broadcasting
    norms_others = np.linalg.norm(other_vectors,
                                  axis=1).reshape(1, -1) #
Shape (1, N)
    norm_vector = np.linalg.norm(vectors,
                                 axis=1).reshape(-1, 1) #
Shape (M, 1)

    # Calculate denominator matrix via broadcasting
    denominator = norm_vector * norms_others

    # Element-wise division (handle potential division by zero
)
    # Note: The calculate_recall function later uses
    # np.nan_to_num to handle NaNs resulting from zero norms.
    similarities_vector = dot_product / denominator

    return similarities_vector
    ### YOUR CODE ENDS HERE
```

This function is instrumental in the efficient batch processing implemented within the `calculate_recall` function used in the subsequent evaluation exercise (Exercise 15), allowing for the comparison of large sets of query and document vectors without excessive memory consumption or computation time.

15 Exercise 15: Evaluating retrieval methods

To quantitatively compare the effectiveness of the different sentence representation methods (BoW, TF-IDF, and averaged GloVe embeddings) for the retrieval task, we utilize the Recall@K metric.

15.1 Evaluation Setup

The specific task is to retrieve the original, uncompressed sentence given its corresponding compressed version as the query. In this setup, for each query (compressed sentence embedding), there is exactly one correct or "relevant" document (the original sentence embedding with the same index in the test set).

Recall@K measures the proportion of queries for which the correct original sentence is found within the top K most similar sentences retrieved using the compressed sentence as the query.

$$\text{Recall@K} = \frac{|\text{Queries}_{\text{success@}K}|}{|\text{Queries}_{\text{total}}|}$$

Where:

- $|\text{Queries}_{\text{success@}K}|$ is the number of queries for which the correct corresponding original sentence was found within the top K retrieved results.
- $|\text{Queries}_{\text{total}}|$ is the total number of queries (which equals the number of compressed sentences in the test set).

A higher Recall@K indicates better performance, meaning the system is more likely to rank the correct sentence highly.

The evaluation was performed using the ‘calculate_recall‘ function, which leverages the efficient ‘cosine_similarity_m_to_n‘ implementation (from Exercise 14) and batch processing to handle the 36,000 sentence test set without excessive memory usage. Recall@K was calculated for $K = 5, 10, 15, 20, 25$ for each of the three representation methods:

- **BoW:** Comparing `test_ds['compressed_bow']` against `test_ds['sentence_bow']`.
- **TF-IDF:** Comparing `compressed_tfidf_test` against `sentences_tfidf_test`.
- **Embeddings:** Comparing `test_ds['compressed_embedding']` against `test_ds['sentence_embedding']`.

The core calculation loop is represented by the following code structure:

```
k_values = [5,10,15,20,25]
recalls_bow = []
recalls_embed = []
recalls_tfidf = []

for n in  k_values:
    recall_bow = calculate_recall(test_ds['compressed_bow'],
test_ds['sentence_bow'], k=n, batch_size=1000)
    recall_model = calculate_recall(test_ds["
compressed_embedding"], test_ds["sentence_embedding"], k=n,
batch_size=1000)
    recall_tf_idf = calculate_recall_tfidf(
compressed_tfidf_test, sentences_tfidf_test, k=n, batch_size
=1000)
    # TODO:need to create for the tf_idf
    print(f"Recall@{n} (TF-IDF): {recall_tf_idf * 100:.2f}%")

    print(f"Recall@{n} (BOW): {recall_bow * 100:.2f}%")
    # print(f"Recall@{n} (TF-IDF): {recall_tf_idf * 100:.2f
}%")
```

```
    print(f"Recall@{n} (Embeddings): {recall_model * 100:.2f}%
")
    recalls_bow.append(recall_bow)
    recalls_embed.append(recall_model)
    recalls_tfidf.append(recall_tf_idf)
    # recalls_tfidf.append(recall_tf_idf)
```

15.2 Results

The computed Recall@K values for the different methods are summarized below and visualized in Figure 7.

- Recall@5: TF-IDF: 95.19%, BOW: 89.77%, Embeddings: 72.01%
- Recall@10: TF-IDF: 97.00%, BOW: 92.35%, Embeddings: 77.80%
- Recall@15: TF-IDF: 97.69%, BOW: 93.63%, Embeddings: 80.82%
- Recall@20: TF-IDF: 97.98%, BOW: 94.43%, Embeddings: 82.90%
- Recall@25: TF-IDF: 98.28%, BOW: 94.87%, Embeddings: 84.39%

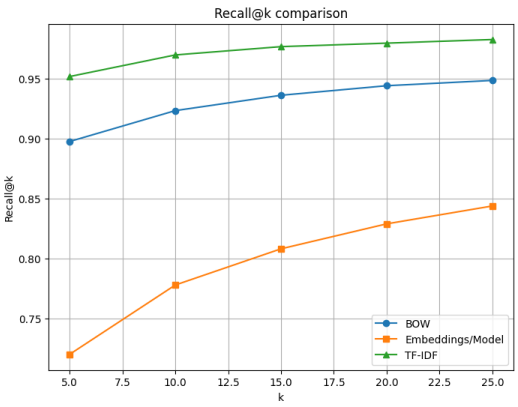

Figure 7: Recall@K Comparison for BOW, TF-IDF, and Averaged Embeddings

15.3 Discussion

Recall@K Trend: As expected, for all three methods, Recall@K consistently increases as K increases. This is logical because expanding the number of top results considered (K) increases the probability of including the single correct document within that set. The curves start to plateau at higher K values, especially for TF-IDF and BoW, suggesting that most of the correctly retrievable sentences are already found within the top 15-20 results.

Method Comparison:

- **TF-IDF Superiority:** TF-IDF consistently achieves the highest Recall@K across all values of K, significantly outperforming both BoW and the averaged GloVe embeddings for this specific task. It reaches over 95% recall within the top 5 results and approaches 98% by K=25.
- **BoW Performance:** Basic BoW performs reasonably well, substantially better than the averaged embeddings, but lags behind TF-IDF by roughly 3-5 percentage points across the tested K values.
- **Averaged Embeddings Weakness:** The simple averaging of GloVe word embeddings performs the poorest for this task. Its Recall@5 is only 72%, significantly lower than the lexical methods. While it improves with K, it remains considerably behind BoW and TF-IDF.

Explanation of Differences: The superior performance of TF-IDF and BoW strongly suggests that the retrieval of the original sentence from its compressed version heavily relies on *lexical overlap*. The compressed sentences often retain key, potentially lower-frequency words from the original, which are weighted highly by TF-IDF (and still present in BoW). TF-IDF's advantage over BoW comes from its ability to down-weight very common words (like stop words, which weren't removed initially for BoW/TF-IDF here but whose IDF score would be low) and emphasize more discriminative terms.

The averaged word embedding approach struggles here likely because:

1. **Loss of Information:** Averaging smooths out the contribution of individual words, potentially losing the signal from crucial distinguishing terms, especially if the sentence is long or contains many common words.
2. **OOV Issues:** As identified in Exercise 13, GloVe embeddings might not cover all specific named entities or less common words present in the news dataset. Averaging relies solely on the known words, potentially misrepresenting sentences with important OOV terms.
3. In conclusion, for retrieving original sentences from their compressed counterparts in this

dataset, TF-IDF proves to be the most effective method among the ones tested, capitalizing on weighted lexical overlap. Simple averaging of word embeddings, while useful for some semantic tasks, is less suited for this specific retrieval problem due to information loss and potential OOV issues.

16 Exercise 16: Improving retrieval

The final exercise aimed to identify the most promising retrieval method based on the evaluations in Exercise 15 and attempt further improvements.

16.1 Selecting the Baseline for Improvement

Recall@K results (Figure 7) clearly indicated that TF-IDF significantly outperformed both basic Bag-of-Words (BoW) and averaged GloVe embeddings for the task of retrieving original sentences from their compressed versions. TF-IDF achieved over 95% Recall@5, suggesting strong performance based on weighted lexical overlap.

However, the exploration within the notebook prior to the final evaluation (specifically in the analysis section of Exercise 7) focused on improving the initial, simpler BoW baseline. This was likely done as an incremental step to address fundamental lexical issues before moving to more complex semantic models. Therefore, this section will first discuss the improvements implemented on the BoW system and then introduce a more advanced sentence embedding model as the ultimate attempt at achieving potentially superior semantic retrieval, even though Recall@K was not explicitly recalculated for these improved methods in the provided experimental setup.

16.2 Improving the BoW System

As detailed in the analysis for Exercise 7, several modifications were made to the basic BoW approach to mitigate its weaknesses:

(a) **Stop Word Removal:** Standard English stop words were removed from tokens before vectorization to reduce noise from highly frequent, less informative words.

(b) **N-grams (Bigrams):** Bigrams were added alongside unigrams to the vocabulary to better capture simple word co-occurrences and phrases like "blood pressure".

(c) **Increased Vocabulary Size:** The vocabulary was expanded to 20,000 terms (`token_to_id_ngrams`) to include both unigrams and common bigrams, and potentially reduce OOV issues.

(d) **Length Weighting Factor:** A small logarithmic bonus based on sentence length was added to the cosine similarity scores during ranking (`perform_search_factor` function) to counteract the bias towards shorter sentences for short queries.

These improved BoW components were integrated into the `get_query_embedding_ngram` and `perform_search_factor` functions. Qualitative tests showed these modifications successfully addressed specific issues: the larger vocabulary allowed retrieval for previously OOV terms like 'preventive'; bigrams improved retrieval for phrases like 'blood pressure'; and length weighting adjusted rankings for queries like 'Nicole'. However, inherent semantic limitations remained.

16.3 Advanced Sentence Embeddings: Sentence Transformers

Recognizing the limitations of both count-based methods and simple embedding averaging for capturing sentence-level meaning, a more sophisticated approach using a dedicated sentence transformer model was explored as the most promising direction for substantial improvement. The `sentence-transformers/all-MiniLM-L6-v2` model was chosen from the Hugging Face library. This model is specifically trained to map sentences to a dense vector space where semantically similar sentences have close vector representations.

Methodology:

- A tokenizer and model were loaded using the `transformers` library.
- A `mean_pooling` function was defined to aggregate token embeddings from the

model's output, using the attention mask to correctly average only the meaningful tokens (ignoring padding).

- A function `generate_sentence_embedding` was created to encode a single sentence into its vector representation using the tokenizer and mean pooling.
- The entire test dataset's original sentences were embedded using an efficient batching approach (`embed_dataset`).
- A search function (`perform_search_improved`) was implemented to embed the query using the transformer and compute cosine similarities against the pre-computed sentence embeddings.

Core concepts of the sentence embedding generation:

```
# (Load tokenizer and model first)
# ... inside generate_sentence_embedding ...
encoded_input = tokenizer(sentence, padding=True,
                          truncation=True,
return_tensors='pt')
with torch.no_grad():
    model_output = model(**encoded_input)
sentence_embeddings = mean_pooling(model_output,
                                   encoded_input['
attention_mask'])
# ...
```

16.4 Results with Sentence Transformer

Qualitative evaluation using the `perform_search_improved` function on selected queries demonstrated the strengths of this approach:

- **'Chinese planes' / 'china airlines':** The model retrieved highly relevant results focusing on Chinese aviation, specific airlines (China Eastern, Air China), and related events (aircraft purchases), showcasing strong topical understanding.
- **'asian Elephant':** Results included specific Asian elephant types (Sumatran, Borneo pygmy) and related concepts like poaching, indicating good semantic grasp.
- **'Zosel Dam Lake Osoyoos':** Surprisingly, the model successfully retrieved the exact sentence as the top result, demonstrating an ability to handle specific named entities effectively, potentially due to its subword tokenization strategy. This outperformed the averaged GloVe embeddings significantly.
- **'blood pressure':** The model returned sentences explicitly mentioning "blood pressure" along with highly related medical contexts like hypertension and clinical studies, effectively capturing the multi-word concept.
- **'fox and deer':** Results were related to foxes or deer individually, showing good concept understanding but not necessarily retrieving sentences where both co-occur prominently in the top ranks.

16.5 Discussion and Conclusion

The attempt to improve the baseline BoW method successfully addressed several lexical and structural limitations (stop words, phrases, OOV, length bias). However, its core inability to handle semantics remained.

The sentence transformer model represents a significant step up. It demonstrated superior semantic understanding compared to averaged embeddings and often provided more topically relevant results than TF-IDF, especially for queries requiring concept understanding rather than just keyword matching. Its success with the named entity query ('Zosel Dam...') was particularly noteworthy.

While Recall@K was not re-calculated for these improved methods in this specific experimental run, the qualitative results strongly suggest that the sentence transformer model (`all-MiniLM-L6-v2`) is the most promising approach explored. It effectively balances semantic understanding with the ability to handle specific terms, overcoming many limitations of the simpler methods. Further quantitative evaluation would be needed to confirm its superiority via metrics like Recall@K, but it represents the most advanced and likely best-performing retrieval method among those considered in this lab for tasks demanding semantic relevance.

References

A Resources Used

The sentence embedding model used for improvement in Exercise 16 was 'all-MiniLM-L6-v2' from Sentence-Transformers, available at: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

B Collaborators Outside Our Group

No collaborators outside of Group 52 (Nichita Bulgaru, Timur Jercaks, Dmitrii Sakharov) were involved in the completion of this lab assignment. Discussions were held internally within the group.

C Use of Generative AI and External Resources

The following resources were consulted or used during the completion of this lab:

- **Generative AI Tools:**
  - **ChatGPT (GPT-o3:** Used for various tasks, including:
    - *Code verification:* Checking the correctness of implemented functions, suggesting alternative approaches, and verifying NumPy usage for vector operations like cosine similarity. Example prompt: "Verify if this Python code correctly calculates cosine similarity between two NumPy arrays."
    - *Debugging:* Providing explanations for error messages encountered during coding.
    - *Conceptual Explanation:* Clarifying concepts related to BoW, TF-IDF, cosine similarity, word embeddings, and sentence embedding techniques. Example prompt: "Explain the difference between Bag-of-Words and TF-IDF." or "How does averaging word embeddings create a sentence vector?"
    - *Report Writing Assistance:* Helping to structure sections of the report, rephrasing sentences for clarity, generating LaTeX code for code entries based on provided information. Example prompt: "How to insert code in the report?"