Homework 1

New Attempt

- Due Feb 5 by 11:59pm
- Points 56
- Submitting a file upload
- File Types pdf, py, and txt
- Available after Jan 22 at 4:30pm

In this assignment, you'll build representations for documents and words based on the bag-of-words model. You'll implement 2 popular weighting schemes for these vectors: tf-idf and PPMI, both discussed in Chapter 6 of the **textbook (https://web.stanford.edu/~jurafsky/slp3/)** . Then you'll compare these weighting schemes on learning word similarity and apply one of them, PPMI, to examine social bias in an NLP corpus.

Learning objectives (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1# objectives)

After completing this assignments, you will be able to:

- Load in text data and manipulate it in Python
- Demonstrate that a model's notions of word similarity comes from the contexts of neighboring words in corpora
- Implement vectors for words, the predominant representation of semantics in NLP
- Investigate how NLP corpora can potentially encode harmful social biases through word associations

Datasets and skeleton code (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1# and-skeleton-code)

Here are the materials to download for use in this assignment:

- **Skeleton Python code (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1/skeleton.py)** . You will need to have a Python environment with scipy and numpy packages. Some functions are stubs in the python code. You will need to fill them out.
- **CSV of the complete works of Shakespeare (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1/shakespeare_plays.csv)**
- **Vocab of the complete works of Shakespeare (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1/vocab.txt)**

- **List of all plays in the dataset (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1/play_names.txt)**
- **List of identity labels (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1/identity_labels.txt)** from **Rudinger et al. 2017 (https://aclanthology.org/W17-1609/)**
- **SNLI corpus (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1/snli.csv)**
  - This corpus is selections from SNLI, a corpus used for the NLP task of “natural language inference” (see **Bowman et al. 2015 dataset paper (https://aclanthology.org/D15-1075/)**). Each line contains a sentence that is either a “premise” (an image caption) or a “hypothesis” produced by annotators to be in a certain logical relation with the associated premise (entailment, neutral, contradiction). You don’t need to worry about these details, but the `sentenceID` column is a unique index for each sentence and `captionID` is an ID for all sentences associated with same caption/premise.

Part 1: Vector spaces (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1# 1-vector-spaces)

Implementation (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#implementa

Open up the `skeleton.py` file. Complete the function stubs using the information below.

Term-document matrix (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#term-document-matrix)

Write code to compile a term-document matrix for Shakespeare’s plays in the function stub `create_term_document_matrix`. Follow the description in the textbook:

> In a *term-document matrix*, each row represents a word in the vocabulary and each column represents a document from some collection. The figure below shows a small selection from a term-document matrix showing the occurrence of four words in four plays by Shakespeare. Each cell in this matrix represents the number of times a particular word (defined by the row) occurs in a particular document (defined by the column). Thus **clown** appeared 117 times in **Twelfth Night**

Here is an example of a term-document matrix (you will build a different one in the homework):

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

The dimensions of your term-document matrix will be the number of documents D (in this case, the number of Shakespeare's plays that we give you in the corpus) by the number of unique word types $|V|$ in that collection. The columns represent the documents, and the rows represent the words, and each cell represents the frequency of that word in that document.

Write this code from scratch. You can use packages like NumPy as helper packages, but do not use packages that directly do this computation for you (like scikit-learn or gensim).

Term-Context Matrix (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#term-context-matrix)

Instead of using a term-document matrix, a more common way of computing word similarity is by constructing a term-context matrix (also called a term-term or word-word matrix), where columns are labeled by words rather than documents. The dimensionality of this kind of a matrix is $|V|$ by $|V|$. Each cell represents how often the word in the row (the target word) co-occurs with the word in the column (the context) in a training corpus. You get to decide what are considered units that form boundaries of that context. For example, do two words have to co-occur within a close window in the same line of the play, or just in the same play? You also can decide when it makes sense for a word to co-occur with itself in the term-context matrix. That is, will the cell for when the same word is target and context always stay 0? Note that there may be vectors where all elements are 0 if a word only appears in documents where it is the only word in the document.

Specifically, implement the `create_term_context_matrix` function. This function specifies the size of the word window around the target word that you will use to gather its contexts. For instance, if you set that variable to be 4, then you will use 4 words to the left of the target word, and 4 words to its right for the context. In this case, the cell represents the number of times in Shakespeare's plays the column word occurs in +/-4 word window around the row word.

Write this code from scratch, i.e. do not use additional packages that directly compute these matrices.

Evaluating vector spaces (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#evaluating-vector-spaces)

So far we have created 2 vector spaces for the words in Shakespeare, one with a dimension of D and another of dimension $|V|$. Now we will try to evaluate how good our vector spaces are. We can do this with an intrinsic evaluation approach by seeing what words within the vocab are most similar to each other/are synonyms with each other and assessing if the output is reasonable.

Implement the `rank_words` function, which will take a target word index and return a list sorted from most similar to least similar using the cosine similarity metric. For the purposes of the assignment, let's just look at the top 10 words that are most similar to a target word between both the term-document matrix and the term-context matrix (with a window size of your choice). Are those 10 words good synonyms? The skeleton code provides an example of using `rank_words` and looking at similar words using the word 'juliet'. It is okay if the top ranked word is the word itself, with a cosine similarity of 1.

Write this code from scratch, i.e. do not use additional packages that directly compute these matrices.

Weighting terms with tf-idf and PPMI (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#weighting-terms-with-tf-idf-and-ppmi)

Your term-context matrix contains the raw frequency of the co-occurrence of two words in each cell and your term-document matrix contains the raw frequency of words in each of the documents. Raw frequency turns out not to be the best way of measuring the association between words. There are several methods for weighting words so that we get better results.

Take your term-document matrix and implement the weighting schemes *Term frequency inverse document frequency (tf-idf)* and *Positive pointwise mutual information* which are defined in Sections 6.5-6.6 of the textbook. These are the function stubs `create_tf_idf_matrix` and `create_ppmi_matrix`.

Questions for the report (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#questions-for-the-report)

1.1. In our term-document matrix, the rows are word vectors of D dimensions. Do you think that's enough to represent the meaning of words? Why or why not?

1.2. Provide the top 10 associated words and cosine similarities (the output from `rank_words`) with `juliet` and at least 2 other target words of your choice for both term-document and term-context vector spaces.

1.3. Just considering the term-document and term-context matrices without any tf-idf or PPMI weighting, which do you think produces similar words that make more sense than others? Why do you think that is the case? Back up your conclusions by referring to the top associated term lists you provided.

1.4. Explain any decisions you made in implementing your functions, such as whether you allowed a target word to co-occur with itself as a context word, and which window size you chose for the term-context matrix. How might any decisions you make impact our results now?

1.5. Provide the top 10 associated words (the output from `rank_words`) with `juliet` and at least 2 other target words of your choice for tf-idf-weighted term-document matrices and PPMI-weighted term-context matrices.

1.6. Compare the ranked word similarities between weighting with tf-idf and using the unweighted term-document matrix. Which do you think produces similar words that make more sense? Back up your conclusions with specific examples.

1.7. Compare the ranked word similarities between weighting with PPMI and using an unweighted term-context matrix. Which do you think produces similar words that make more sense? Back up your conclusions with specific examples.

1.8. Overall, do some approaches appear to work better than others, i.e produce better synonyms? Do any interesting patterns emerge? Discuss and point to specific examples.

Part 2 (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1# 2)

In this part, you will measure associations between words in a commonly used NLP corpus, SNLI, and comment on the potential for encoding problematic social biases.

Implementation (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#implementa 1)

There is no skeleton code for this section, but you can reuse code from Part 1.

First, write a loader for text from the SNLI corpus into a similar format as the Shakespeare corpus was loaded. You can use the `sentenceID` column as the document name, though this will not be as important since you'll only be building a term-context matrix. You'll still want to tokenize and lowercase the input as was done with the Shakespeare corpus. It is also encouraged to remove stopwords (such as with a list from the NLTK package) and remove low-frequency terms from the vocabulary.

Then, build a term-context matrix in a similar fashion as with the Shakespeare corpus and apply PPMI weighting. You can choose the size of the context window. You can also use the whole sentence as the context. If this matrix is too big or taking too long to calculate, filter to just words that occur over some frequency threshold in the entire corpus.

Find words associated with identity labels in SNLI (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#find-words-associated-with-identity-labels-in-snli)

Now you will use the list of identity terms provided above in the 'Datasets and skeleton code' section. Examine which words are highly associated with selected identity labels in the SNLI corpus with PPMI using the `rank_words` function. Choose identity labels that are related, such as multiple terms for gender, multiple terms for race/ethnicity, or other relations. Look for any associations that may reflect social stereotypes or possible *representational harms* in machine learning, defined below from Blodgett et al. 2020 (https://aclanthology.org/2020.acl-main.485/) :

> Representational harms arise when a system (e.g., a search engine) represents some social groups in a less favorable light than others, demeans them, or fails to recognize their existence altogether.

Types of representational harms from Blodgett et al. 2020 (https://aclanthology.org/2020.acl-main.485/) include:

> 1. Stereotyping that propagates negative generalizations about particular social groups

> 2. Differences in system performance for different social groups, language that misrepresents the distribution of different social groups in the population, or language that is denigrating to particular social groups.

Qualitative analysis of word contexts (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#qualitative-analysis-of-word-contexts)

Now you will explore the contexts in the dataset that lead to high PMI association with context words, especially for any words that show social bias (if you found any). *1st-order similarity* is when a target word (in this, case, an identity label) occurs in the same document with a top-associated term. This might not be very informative to see how these words are related in the dataset. If not, look at *2nd-order similarity*, in which the two words occur with similar context words. This can also be examined by looking at the target vectors for the identity term and the highly associated other term in the term-context matrix. These vectors may share high values in dimensions that correspond to certain context words.

Questions for the report (https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#questions-for-the-report-1)

2.1 Provide the top 10 associated context words (by PPMI, in the SNLI corpus) for at least 4 identity labels of your choice. Choose identity labels that are related, such as multiple terms for gender, multiple terms for race/ethnicity, or other relations.

2.2. Do you see any associations learned by this bag-of-words model on the SNLI corpus that be representational harms, such as negative social stereotypes? Compare the top PPMI words for certain identity terms with other related ones (such as men compared with women). Discuss and provide selected results. If you don't find any representational harms (that's okay), provide examples of what you examined and how you interpreted those associations. If you do find problematic associations, specify how they could be harmful.

2.3. For at least 4 pairs of identity terms and highly associated words, provide the document contexts in the SNLI dataset that contribute to this association. Provide actual sentences from the SNLI corpus where either:

- an identity term occurs together with the associated word you found (1st-order similarity) or
- an identity term occurs separately from the associated word, but occurs with similar context words (2nd-order similarity). You'll want to compare values of dimensions in the vectors for both words in this case.

Discuss any findings, particularly related to any associations you found to represent harmful stereotypes.

Deliverables

(https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#

- Your report with results and answers to questions in Part 1 and Part 2, named `report_{your pitt email id}_hw1.pdf`. No need to include @pitt.edu, just use the email ID before that part. For example: `report_xianglli_hw1.pdf`.
- Your implementations for the functions in the skeleton code `hw1_skeleton_{your pitt email id}.py`. You are welcome to put code for Part 2 in the same or a different file. If it's different, please where it is in the README.txt.
  - You are welcome to import any packages you need but please don't modify the function that has already been implemented.
- A README.txt file explaining
  - how to run your code
  - the computing environment you used; what programming language you used and the major and minor version of that language; what packages did you use in case we replicate your experiments (a `requirements.txt` file for setting up the environment may be useful if there are many packages).
  - any additional resources, references, or web pages you've consulted
  - any person with whom you've discussed the assignment and describe the nature of your discussions
  - any generative AI tool used, and how it was used
  - any unresolved issues or problems

Please submit all of this material on Canvas as individual files. Do **not** submit a zip file. Only files with .pdf, .txt, or .py file extensions will be accepted. If you used Jupyter Notebook to complete the assignment, please download it as a .py script. We will grade your report and look over your code.

Acknowledgments

(https://michaelmilleryoder.github.io/cs2731_fall2024/hw1#

This assignment is adapted from Prof Michael Yoder, Prof. Diane Litman and Prof. Mark Yatskar, as

Homework 1

Criteria	Ratings		Pts
Part 1 code completion Completed all 5 functions	**8 pts Full Marks**	**0 pts No Marks**	8 pts
Part 1 code: specified any necessary packages, Python version	**3 pts Full Marks**	**0 pts No Marks**	3 pts
Part 1 results are reasonable (top associated words with at least 'juliet') Some variation is allowable, but there is an issue if similarity numbers are very different from the reference code and/or closest words seem not reasonable	**8 pts Full Marks**	**0 pts No Marks**	8 pts
Part 1 report: discusses whether term-document matrix dimensions are enough	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Part 1 report: lists any decisions when implement functions	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Part 1 report: makes at least 2 comparisons among term-context, term-document, tf-idf, PPMI for multiple terms. Speculates why these differences may occur	**15 pts Full Marks**	**0 pts No Marks**	15 pts
Part 2: code (appropriate loader and PPMI-weighted term-context matrix	**5 pts Full Marks**	**0 pts No Marks**	5 pts
Part 2 report: gives top associated words for at least 4 identity labels	**5 pts Full Marks**	**0 pts No Marks**	5 pts
Part 2 report: discusses if they found any social stereotypes or representational harms with identity labels	**4 pts Full Marks**	**0 pts No Marks**	4 pts
Part 2 report: provides examples of contexts where stereotypes were associated, or what they looked at that they judged as not stereotypical	**4 pts**	**0 pts**	4 pts

Criteria	Ratings		Pts
	Full Marks	**No Marks**	
			Total Points: 56