Programming Assignment 1

Word Embeddings and Named Entity Recognition

Deadline: 14/02/2026 11:59PM

Submission link: https://forms.gle/ERKXaqsEgeavGCQc6

In this assignment, we will investigate how word representations are shaped by different algorithms (SVD vs. GloVe), their context window, and their source. We will bridge the gap between unsupervised semantics (distributional) and supervised prediction, while providing a direct comparison between feature engineering and feature learning. Finally, we will evaluate these vectors against traditional lexical features in a CRF-based Named Entity Tagger.

Task 1. Glove pre-training

The first objective is to implement the GloVe objective function to learn vectors from a CC-News subset. We need to build a vocabulary V of unique tokens (of size N). Using the corpus, construct a global word-word co-occurrence matrix X of size $N \times N$, where X_{ij} represents the number of times word j appears in the context of word i.The model learns an embedding matrix of size $N \times d$, where d is the embedding dimension.The model minimizes a weighted least-squares loss:

$$L = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

The weighting function $f(X_{ij})$ is defined as $(x/x_{max})^\alpha$ if $x < x_{max}$, and 1 otherwise. This prevents rare co-occurrences from being noisy.

Parameters to Set:
Context Window (w): The number of words to the left or the right of the target word.
Embedding Dimension (d): The length of the resulting word vectors.
Weighting Hyperparameters: Specifically x_{max} and α (commonly set to 100 and 0.75 in the paper).
Optimization: Learning rate and number of iterations for the gradient descent process.

Note: To avoid an explosion of runs for each hyper-parameter, we suggest first fixing the dimension at d=200 and optimize the remaining hyper-parameters. Once the optimal configuration is identified, use those fixed parameters to fetch the d-dimensional embeddings for each $d \in \{50, 100, 200, 300\}$.

Task 2. SVD pipeline

Unlike the iterative nature of GloVe, the SVD approach finds the best low-rank approximation of the term-document matrix Z. The matrix Z is decomposed into $U\Sigma V^T$ and truncated to rank d.

Parameters to Set: The primary parameter here is the selection of the top-d singular values and their corresponding singular vectors. We test SVD on the same set of $d \in \{50, 100, 200, 300\}$ as in Task 1.

Generating Token Representations: To get the final token-wise representations for each value of d, we will use the product: $W_d = U_d \Sigma_d$

These resulting $N \times d$ matrices [total of 4] will be used in Task 4. to empirically determine which embedding dimension provides the best signal for the NER task

Task 3. NER using CRF

Coming to the first supervised learning task, we will use the CoNLL-2003 dataset to train a Conditional Random Field model. Here, the focus is on feature engineering. There are 9 distinct NER tags to detect(e.g., B-PER, I-ORG, B-LOC).
Beyond the words themselves, we need to create a robust feature set. This can include:

- Lexical: Current word, surrounding window (prev/next).
- Shape: Capitalization patterns, presence of digits.
- Sub-word: Prefix/Suffix patterns

The goal is to see how much performance we can get out of these hand-crafted features before moving to neural methods.

Note: No need to build CRF from scratch, you can use libraries such as `sklearn-crfsuite`

Task 4. NER using feature learning

We move from feature engineering to feature learning. Instead of using the hand-crafted features from Task 3, we will build a simple Neural Network (MLP) that takes only a single token's embedding as input and predicts its NER tag.

Now, train different MLPs for each dimension d and each algorithm (GloVe and SVD). This allows to directly compare the feature learning capability of unsupervised embeddings across different embedding sizes against the manual features of the CRF.

Task 5. [Extra Credit] SVD Performance Boost

Once you find your best SVD d in Task 4, apply a TF-IDF transformation to the term-document matrix before running the SVD.

- Quality Check 1: Pick 5 diverse words (ex: a noun, verb, city name). Compare the top-5 nearest neighbors of the Raw SVD vectors vs. the TF-IDF SVD vectors. Observe any changes in the top-5 list.
- Quality Check 2: Train one final MLP model using these TF-IDF SVD vectors at the best d. Report if this "weighted" approach outperforms the raw SVD results from Task 4.

Evaluation Protocol:

NER Classification: Use the splits as provided in the CoNLL-2003 dataset.
Source: CoNLL-2003
For obtaining word embeddings use this dataset that we created from the CC News dataset. The provided dataset is a json file where keys are vocabulary and values are list of index (0 based) and corresponding passages from the CC News that contains the vocabulary.

For the unsupervised training tasks (Task 1 and Task 2), we are providing two resources to ensure consistency:

a. Vocabulary: list of N (~25k) tokens derived from the NER dataset.
b. Filtered CC-News: A subset of approximately ~67k documents (**In CC News every instance consists of title and article, so by document we mean title + article**).

Note: GloVe and SVD training must be performed only on the provided vocabulary.
Note: Consider words with upper and lower case as two different words. Ex - Treat words "While" and "while" as two different words.
Handling Out-of-Vocabulary (OOV) Tokens: When evaluating the CoNLL-2003 dataset in Task 4., you may encounter tokens that are not present in vocabulary, need to implement an OOV strategy. Choose your approach wisely ,and justify it in your report. Some suggestions would be : Sub-word Averaging, creating a special <UNK> token, heuristic based longest prefix/suffix match.

Mention in report:

- Report final metrics on the test set provided by CoNLL-2003.
- Justify the OOV strategy used.

Task 1:

- List your final choice for the context window, learning rate, and iterations(any other hyper-parameter) that was tested on fixed d.
- Provide visualizations of the loss curves for $d \in \{50, 100, 200, 300\}$. Analyze latency and loss values.
- List the top-5 nearest neighbors for any 3 chosen words in the vocab.

Task 2:

- List the top-5 nearest neighbors for the same 3 chosen words of Task1.

Task 3:

- List every feature included in your CRF and identify which ones were most important.

Task 4: Describe MLP architecture.

- Provide a table showing Token-level Accuracy, Macro-F1 Score for every version of MLP: GloVe-MLP and SVD-MLP at $d = \{50, 100, 200, 300\}$
- State the best overall configuration. Compare this against CRF. Discuss why one outperformed the other.

Appendix

- For the NER dataset, install the dataset library. Ex. using `pip install datasets==3.6.0`
- We recommend using scipy.sparse.csr_matrix for storing the sparse term-document matrix in Task 2.
- For the SVD decomposition, use Truncated SVD implementations.

GenAI Usage Policy: Any use of LLM assistants such as Claude, ChatGPT, Gemini, etc. needs to be documented in full length in your reports. You will also have to provide shareable links to all chats which you used to implement this programming assignment. Using these assistants and not providing this information goes against the course's honor code.

Please note that the goal is for you to develop a deep as well as big picture understanding of the assignment and your code. You are free to use AI tools to achieve this. However, if you just dump the entire assignment into an AI tool and expect everything to just work, this often will not be the case. If you use the tool for 100% of your project without thinking, this is often directly visible in the code quality and the understanding in the report and we will account for it. TL;DR: we don't care if you used ChatGPT etc, but we do care if it was done blindly.

However, if you use these tools but don't declare that you used them, that again is visible in the code/report, and will incur a zero.

Submission Protocol:
Moodle will be used for the submission of the assignment.

- All the findings need to be put down into a report file.
- Each task needs to be in a separate task_<number>.py file. You can add the common functions in utils.py file. Do not create a single file with all the tasks.
- **Do not add any jupyter notebook.**
- Zip all the code and the asset files, if any, along with the report into a single file and then submit it on Moodle. Name the compressed file as [Roll1_roll2.(zip/tar.gz)]
- If the code or assets does not fit in the Moodle submission limit, they can be offloaded onto Google Drive and working links provided in the report.

Note - Only one member of the team should submit the assignment on Moodle. **Teams can have up to four people**.