Natural Language Processing Assignment 1

February 25, 2025

Due date: 3/18 11:59 PM. These questions require thoughtful responses, but concise answers are expected. While group discussions are encouraged, the final work must be completed independently. Please submit a package consisting of a report (in LaTeX or Markdown) and source code with necessary annotations. Your report should include your implementation ideas, observations, and must not exceed 8 pages. Ensure the report contains your name and student number. You do not need to submit the dataset. If you have any questions, feel free to discuss them during office hours in the WeChat group NLP@THU2025.

1 Word2Vec (10 points)

1.1 Word2Vec Implementation (7 points)

In this part, you will implement a Word2Vec model and train your word embeddings using a deep learning framework such as TensorFlow or PyTorch. Your model should be based on either the Continuous Bag of Words (CBOW) or Skip-gram architecture, with Negative Sampling.

Tasks

- **Implementation:** Implement Word2Vec using either CBOW or Skip-gram. You are allowed to use Python libraries such as `gensim`, `nltk`.
- **Ablation:** Experiment with embedding dimensions (100, 300, 500).
- **Dataset:** Train your model on a subset of Wikipedia data. Use the Wikipedia dataset. Since the full dataset is too large, extract a smaller portion for training. Perform basic preprocessing (e.g., lowercasing).
- **Evaluation:** Evaluate embeddings using the WordSim353 dataset by calculating Spearman's correlation coefficient. Compute the Spearman's correlation coefficient between your embeddings and the WordSim353 dataset.

1.2 Word2Vec Improvement (3 points)

In this section, you are required to improve the performance of the Word2Vec model from the previous problem using various techniques. Potential methods for improvement include:

- Incorporating word sense disambiguation
- Leveraging external word knowledge sources
- Evaluating character-level embeddings

You should at least implement one improvement. After applying these improvements, you should reevaluate the resulting embeddings using the WordSim353 dataset and analyze any changes in performance. The evaluation should cover both the idea behind your chosen techniques and the observed performance improvements. In addition to WordSim353, you are encouraged to explore other evaluation metrics such as the SCWS (Similarity and Relatedness Benchmark). To further enrich your experiments, consider visualizing the embeddings and comparing the results across different configurations.

For incorporating external word knowledge, you may utilize a lexical database such as WordNet [1] or HowNet HowNet3. WordNet can be accessed through the NLTK package, following the API instructions provided at nltk_wordnet.

2 Pretrained Model Embedding Generation (5 points)

In this part, you will use a pretrained small language model from Huggingface to generate word embeddings. Due to resource constraints, you are encouraged to use local or online platform resources such as Google Colab. The model you choose should be small enough to run on the platform. The following models are recommended:

- **openbmb/MiniCPM-1B-sft-bf16** [2]
- **TinyLlama/TinyLlama_v1.1** [3]
- **Qwen/Qwen2.5-0.5B** [4]

A provided code snippet in `reference_assignment1.py` explains how to extract these embeddings. Select one of these models and extract embeddings from the input embedding matrix or the hidden states of the model.

3 Evaluation (3 points)

Evaluate the generated embeddings using two tasks:

- **Word Similarity Task**: This task assesses the similarity between words. Use the WordSim353 [5] dataset, which contains 353 word pairs with human-annotated similarity scores. Compute the Spearman's rank correlation coefficient for the Word2Vec and Small Language Model embeddings. The dataset `wordsim353_agreed.txt` is available from WordSim353.

- **Paraphrase Detection Task**: In this task, you will use the embeddings (either word or sentence embeddings) to predict paraphrases. The task involves identifying whether a pair of sentences are paraphrases of each other. After obtaining the similarity score between the sentences, classify the pair as paraphrases if the score is greater than or equal to a predefined threshold (which should be fixed for the task), or as non-paraphrases if the score is below the threshold. Report the threshold and accuracy of each model on this task. Use the dataset `msr_paraphrase_test.txt`, which is available from Microsoft Research Paraphrase Corpus [6].

4 Discussion and Analysis (2 points)

In this section, compare the performance of the Word2Vec embeddings with those generated by the small language model. Analyze the strengths and weaknesses of each approach. Discuss any challenges encountered during implementation and evaluation.

5 Scoring Breakdown

- Word2Vec Implementation: 7 points
- Word2Vec Improvement: 3 points
- Pretrained Model Embeddings: 5 points
- Evaluation (WordSim353 + Paraphrase Detection): 3 points
- Discussion and Analysis: 2 points

References

[1] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[2] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[3] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

[4] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[5] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414, 2001.

[6] William Dolan, Chris Quirk, Chris Brockett, and Bill Dolan. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. 2004.