CGE096 - Natural Language Processing

What is this course about?

Natural language processing (NLP) is a branch of Artificial intelligence. Modern approaches are now available to solve problems in web search, question answering, language translation, finding the contextual meaning of a word, sentiment analysis, textual entailment, coreference resolution, named entity recognition, etc.

Many NLP tasks have gained performance improvement by several folds using deep learning approaches in the last decade.

In this course, students will gain a foundational understanding of NLP and deep learning neural network models to solve NLP tasks. They will also gain enough hands-on experience to process a large corpus and solve some NLP tasks.

Through lectures and assignments, students will learn the necessary skills to process a large corpus and apply techniques to estimate word vectors, design and implement language models, etc., using neural networks.

Course outline

- Introduction to NLP
- Counting and empirical laws
- Managing and processing a large corpus (> 5GB)
- Word Representation using a vector
- Word vectors, contextual word embedding
    - Traditional methods - HAL, COALS, SVD
    - ANN Models - Word2Vec, Glove, ELMO
- Language Generation/Language Models
    - Traditional n-gram models
    - RNN (GRU and LSTM)
    - BERT
- Language Translation
    - IBM Models I and II
    - Encode and Decoder models using RNN
    - Attention mechanisms
    - Transformers
- ~~Chat-bots~~
- ~~Question Answering System~~

Some Assignment samples

1. The partial data set, sourced from Kaggle, contains research articles related to various specializations related to COVID-19. This corpus has around 56000+ files. In this assignment, you must perform a set of tasks from the JSON-encoded partial COVID-19 dataset as given below.

    - Extract the text content from the JSON-encoded data set and create a text corpus.
    - Develop your preprocessing steps and order of steps
    - Count the frequency of the word in the vocabulary and compute its corresponding rank. Using this table, find the average value of alpha
    - Plot Tokens Vs Vocabulary graph using Heaps' empirical law. Find Vocabulary count for every 10000 tokens. You may use a log scale for plotting

2. Implement the COALS algorithm in Python and estimate the word vectors for the dataset COVID19.
    - Use SVD to compute a word vector of size 50
    - Use PyTorch implementation of Word2Vec to compute the word-vectors.
    - Initialize the input embedding matrix with the word vectors estimated using COALS-SVD. Make sure that the vocabulary used in the COALS-SVD and Word2Vec are the same.
    - Select ten words (related to COVID infection) and list their syntactic (associative) and semantic similarities. Show at least five words each for syntactic and semantic similarities along with their cosine distance score

3. Finding similarities in sentences (Assignment weight 40%). In this assignment, you will find the sentence similarities using an RNN. The input to the RNN is a variable-length sentence. You will use the word embedding as input at each time step. You will use BPTT to learn the embedding.
    - The output should be a vector whose length should be at least the size of the word vector (maximum sentence vector size should be 1.5 times the size of the word vector)
    - Decide on the similarity measure and explain (at least 100 words) why you had selected the distance measure.
    - Show at least five example sentences of similarities. You will Display ten similar sentences and their corresponding similarity score for every example you have chosen.
    - Use the COVID-19 corpus to complete the assignment

4. In this assignment, you will build a knowledge graph using similar words obtained from Word2Vec. Consider the same COVID19 corpus for this purpose.
    - Step 1: Preprocessing
        - You need to preprocess the corpus before building word vectors. This is the most important step to build optimal word vectors
    - Step 2: Build Word vectors
        - You may choose to use any popular package to build the Word vectors.
    - Step 3: Selection of keywords
        - Pick a list of keywords from the vocabulary (up to 10). For every keyword in the list find the k-nearest neighbors or similar words (order by proximity) iteratively. The sample code is two levels deep. For a good knowledge graph, you may go up to 4 levels
    - Step 4: Build a graph
        - The next important piece of the puzzle is to find how well a keyword is connected with other keywords. Here, we want to use the Graph to structure the connected keywords. It allows us to represent the connections in a structured way. It is also possible to filter the connections by studying the edges of the nodes (representing the relationship of the keyword) and the nodes representing the keywords. Please refer to the "networkx" library for a detailed explanation

References

1. Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft)
2. Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing
3. Lund, K., and C. Burgess. "Hyper-space analogue to language." Behavior Research, Instruments, & Computers (1996).
4. Douglas L. T. Rohde, Laura M. Gonnerman, David C. Plaut, “An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence”, Communications of the ACM, 2006
5. Deerwester, Scott, et al. "Indexing by latent semantic analysis." Journal of the American Society for information science 41.6 (1990): 391-407
6. Tomás Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, ”Efficient Estimation of Word Representations in Vector Space”, 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013. 1301.3781.pdf (arxiv.org)
7. Pennington, J.; Socher, R. & Manning, C. D., “GloVe: Global Vectors for Word Representation,” *Empirical Methods in Natural Language Processing (EMNLP),* **2014**, 1532-1543
8. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the

Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pages 2227–2237

9. Sepp Hochreiter and J¨urgen Schmidhuber. “Long Short-Term Memory”. In: Neural Comput. 9.8 (Nov. 1997), pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. url: http://dx.doi.org/10.1162/ neco.1997.9.8.1735
10. Brown, Peter F., et al. "A statistical approach to machine translation." Computational linguistics 16.2 (1990): 79-85.
11. Brown, Peter F., et al. "The mathematics of statistical machine translation: Parameter estimation." Computational linguistics 19.2 (1993): 263-311.
12. Koehn, Philipp, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. University of Southern California Marina Del Rey Information Sciences Inst, 2003.
13. Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
14. Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in Neural Information Processing Systems 13 (2000).
15. Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).
16. Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. 2018 Oct 11.
17. Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017)
18. Hori, Chiori, and Takaaki Hori. "End-to-end conversation modeling track in DSTC6." arXiv preprint arXiv:1706.07440 (2017).