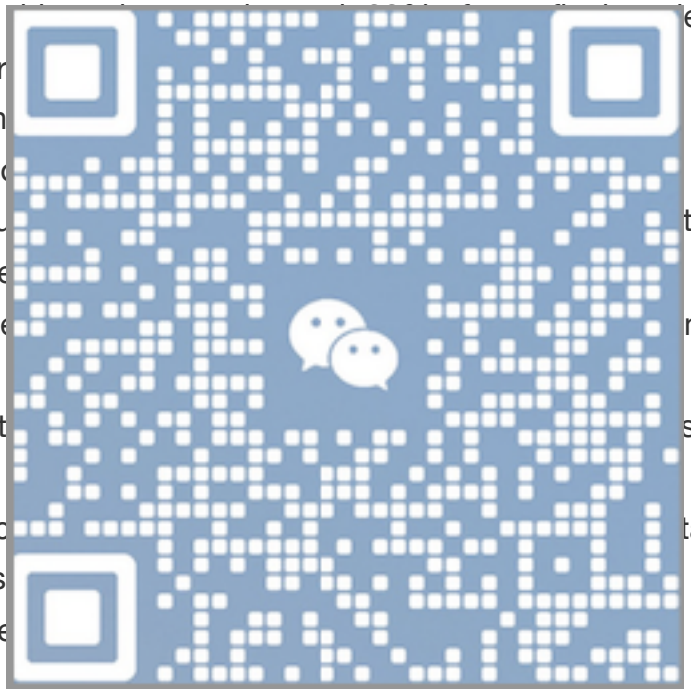
CSC 485H/2501H: Computational linguistics, Fall 2024 - Assignment 2

Due date: 17:00 on Thursday, November 7, 2024.

General Instructions

- Late assignments will not be accepted without a valid medical certificate or other documentation of an emergency.
- For CSC485 students, [illegible]e. For CSC2501 students, it is worth 25% of your
- Read the whole assign
- Type the written parts
- Your work must be you [illegible] the problems. If you need assistance, contact the
- Any clarifications to the [illegible]m for the class. Check the page regularly.
- The starter code direct [illegible]s. Refer to code files in that directory.
- When implementing co [illegible]tant instructions, implementation details
- Fill in your name, stude [illegible] the top of each file you submit.



0. Warming up with WordNet and NLTK (4 marks)

(a) Deepest function (1 mark)

- Implement the `deepest` function in `q0.py` to find the synset in WordNet with the largest maximum depth and report both the synset and its depth on each of its paths to a root hyperonym. Hint: use `wn.all_synsets` and `synset.max_depth` methods.

(b) Superdefn function (2 marks)

- Implement the `superdefn` function in `q0.py` that takes a synset `s` and returns a list consisting of all of the tokens in the definitions of `s`, its hyperonyms, and its hyponyms. Use `word_tokenize` as shown in chapter 3 of the NLTK book.

(c) Stop_tokenize function (1 mark)

- Implement the `stop_tokenize` function in `q0.py` that takes a string, tokenizes it using `word_tokenize`, removes any tokens that occur in NLTK's list of English stop words and also removes any tokens that consist entirely of punctuation characters. Use Python's punctuation characters from the string module. Maintain the original case in the return value.

1. The Lesk alg arks)

(a) Mfs function (

- Implement the `mfs` fu a given word in a sentence. Note that `wordnet.syn` ency.

(b) Lesk function

- In the `lesk` function i m as specified in Algorithm 1, including rns the cardinality of the intersection of the bag ze function to tokenize the examples and definitio

(c) Lesk_ext function (3 marks)

- In the `lesk_ext` function in `q1.py`, implement a version of Algorithm 1 where the `signature` also includes the words in the definition and examples of sense's hyponyms, holonyms, and meronyms. Use `stop_tokenize` as before.

(d) Justification for lesk_ext (2 marks)

- Explain why the extension in `lesk_ext` is helpful. Consider the likely sizes of the overlaps.

(e) Lesk_cos function (4 marks)

- In the `lesk_cos` function in `q1.py`, implement a variant of `lesk_ext` that uses `CosSim` instead of `Overlap`. Modify `signature` and `context` to be vector-valued and construct the vectors as described. Use `stop_tokenize` to get the tokens for the signature.

(f) Lesk_cos_oneside function (2 marks)

- In the `lesk_cos_oneside` function in `q1.py`, implement a variant of `lesk_cos` that, when constructing the vectors for the signature and context, does not include words that occur only in the signature. Use `stop_tokenize` to get the tokens for the signature.

(g) Comparison of lesk_cos_oneside and lesk_cos (3 marks)

- Compare how well `les` `cos`. Justify your answer with examples.

(h) CosSim with b

- If we use `CosSim` for v ow is it related to the set intersection? (No imple

(i) Lesk_w2v func

- In the `lesk_w2v` funct where the vectors for the signature and context a 2vec vectors for the words in the signature and se ntext as sets rather than multisets. Use `stop_tokenize` to get the tokens for the signature.

(j) Lowercasing tokens (2 marks)

- Alter your code so that all tokens are lowercased before they are used for any of the comparisons, vector lookups, etc. Analyze how this alters the different methods' performance and explain why. Do not submit this lowercased version.

2. Word sense disambiguation with BERT (22 marks)

(a) Context necessity (4 marks)

- Is context really necessary? Give an example of a sentence where word order–invariant methods such as those implemented for Q1 will never be able to completely disambiguate. Explain the more general pattern and why these methods cannot provide the correct sense for each ambiguous word.

(b) Gather_sense_vectors function (10 marks)

- Implement `gather_sense_vectors` in `q2.py` to assign sense vectors as described.

(c) Sorting corpu marks)

- In the docstring for `ga` orpus by length before batching is much faste .

(d) Bert_1nn func

- Implement `bert_1nn` entence given sense vectors produced by `g` n the docstring about loop usage.

(e) Issues with ar

- Think of at least one ot to use the code for this assignment to disambiguate arbitrary sentences. Consider either the Lesk variants from Q1 or the BERT-based method here (or both).

3. Understanding transformers through causal tracing (16 marks)

(a) Get_forward_hooks function (3 marks)

- Implement `get_forward_hooks` .

(b) Causal_trace_analysis function (5 marks)

- Implement `causal_trace_analysis` to compute the impact of states, MLP and attention.

(c) Causal tracing result report (1 mark)

- Report your generated causal tracing result plots for the prompt “The Eiffel Tower is located in the city of” with the output “Paris” in your report.

(d) Model size impact on causal tracing (3 marks)

- Experiment with different sizes of GPT-2 models (e.g., small, medium, large, and XL) to examine how model size impacts causal tracing patterns. Address the following in your report:
  - At what model size do you observe that the causal tracing pattern no longer appears?
  - Discuss potential r tracing patterns occurs as the model size inc

(e) Prompt types erns (2 marks)

- Using GPT-2 XL, exper ypes that result in a causal tracing pattern similar r findings with examples and discuss what char is similarity.

(f) Absent or dimi s (2 marks)

- For GPT-2 XL, explore the causal tracing pattern is absent or significant othesize why the pattern does not emerge. Discuss any trends or patterns you identified and reflect on the broader implications of how language models process, store and generate factual information obtained from pretraining.

What to submit

- Submit electronically via MarkUs.
- Submit a total of five required files:
  - `a2written.pdf` : a PDF document containing answers to questions 0a, 1d, 1f, 1h, 2a, and 2d. Also include a typed copy of the Student Conduct declaration and sign it by typing your name.
  - `q0.py` : the entire file with your implementations filled in.