DD2417 Language Engineering
Project topics

Johan Boye
2024-04-30

Please select one of the topics below. **You are welcome to come up with your own topic**, but if you do that, then please first mail Johan `<jboye@kth.se>` to discuss the scope of that project.

A general comment: Many of these problems can be solved by prompting ChatGPT, Gemini, or a similar Transformer-based chatbot. **This will not be considered a solution**, although it can be interesting to *also* use ChatGPT to be able to compare its performance with that of your own implementation.

1. (Text summarization) It is obviously very useful to be able to summarize a text automatically. Early research on text summarization focused on *extractive summarization*, where sentences from the original text are selected and assembled to form the summary, which usually does not lead to very good summaries. The *abstractive* approach, where the summary is written from scratch, is more challenging but has become realistic with the development of the new neural-network-based methods. In this task, we ask you to write a text summarizer, **either** by implementing a RNN-based approach with attention, **or** by using one or several pre-trained transformer models which are then fine-tuned on a summarization dataset. Two available datasets are WikiHow and Amazon Fine Food Reviews, but there are several others as well. The results can be evaluated using some established metric like ROUGE and (to a certain extent) manually.

2. (Word predictor) Most modern mobile phones have some kind of word prediction software. As you are typing a word in a text message or an e-mail, the word predictor displays a shortlist of the most probable completions of the word. The list of suggestions is updated for each keystroke the user makes. If one of the suggestions indeed is the word you intended to type, you can type the word just by clicking on it, thereby saving many keystrokes.

   Such a system must have a language model to be able to suggest the next word. The task in this project is to

   - implement word prediction using two different language models, one based on n-gram models, and one based on neural networks. The neural-network-based model can be implemented using a feed-forward network, similar to what we did in the exercise to assignment 4 (see also the description in section 7.5 in the textbook and lecture 8a), and/or an RNN as described in section 9.3 in the textbook (and lecture 8b), and/or a model which you construct yourselves based on the transformer architecture.
   - implement some simple GUI to show how your word prediction works
   - evaluate by computing the proportion of saved keystrokes using different models and inputs

3. (Headline generation) Given a news item, what would be the best headline? You might try to fine-tune an existing language model to generate headlines from news items, using this English data set and/or this Swedish data set.

4. (Pairing news items and headlines) As a simpler(?) variant of the above task, one might try to separate news stories from their headlines and then try to pair them up again. Use the data set(s) from the previous topic.

5. (Semantic equivalence detection) People tend to ask many questions on Internet forums like Reddit or Quora. Some questions tend to ask the same thing, even though they are not expressed using exactly the same words. Here is a dataset containing some 400,000 pairs of questions with equivalence annotations: "1" means that the two questions are equivalent, "0" means that they are not. Using this dataset and some machine learning methods (at least one method based on neural networks and one method based on some other principle), train and evaluate classifiers that can determine whether or not two questions are semantically equivalent (i.e. they ask for the same thing).

6. (Nested Named Entity Recognition) In the assignments, we studied the task of identifying *named entities* in a text, the so-called "Named Entity Recognition" (NER) problem. An example of a named entity could be a phrase like "Bank of England". However, an observant reader would say that this phrase actually contains *two* named entities: "Bank of England" (an organization), and "England" (a country). A program able to detect *all* named entities, also those contained in other named entities, is solving the task of "Nested Named Entity Recognition" (NNER).

   Your task is to implement a program that can solve the NNER problem. To your help, you have the GENIA dataset, and you might also get some inspiration from the following papers (although you are free to solve the problem using whichever method you like):

   - Nested Named Entity Recognition (EMNLP 2009)
     https://aclanthology.org/D09-1015.pdf
   - A Neural Layered Model for Nested Named Entity Recognition (NAACL 2018)
     https://aclanthology.org/N18-1131.pdf
   - Named Entity Recognition as Dependency Parsing (ACL 2020)
     https://aclanthology.org/2020.acl-main.577.pdf

7. (Question word prediction) Question answering (QA) is a very interesting and well-studied NLP task, where given a text and a question, the task is to find the correct answer. A less well-studied but also interesting task is Question word prediction (QWP), i.e. finding a correct question word (or phrase), given the rest of the question and the correct answer. For instance, if you have a question "What is the capital of Sweden?" with the answer "Stockholm", your QWP model will get the following question-answer pair (QA-pair) as an input:

   Question: `<qw> is the capital of Sweden?`
   Answer: Stockholm

   The output for this QA-pair will then be "What". Note that the output will not always be one word, for instance "How many" or "How much" are also valid question phrases. Your task is to train and evaluate a QWP model using any available QA-corpus, for instance, the SQuAD corpus.