Natural Language Processing: Final project

Reykjavik University – School of Computer Science

Instructors: Stefán Ólafsson, Hannes H. Vilhjálmsson

September 2023

1 Description

Your goal in the final project is to develop a working NLP system. At the end of the course you will demonstrate your system as well as hand in a report describing your work (see Section 2). Note that you will need to read some research papers (and refer to them in your report) in order to acquire the necessary background. The course instructors can assist you in selecting the appropriate papers.

It is assumed that your system uses any of the techniques discussed in the course. **It is preferred that you work on this project in a group of 2-3 students**.

Below are some project ideas, but note that **you are allowed, and encouraged, to propose your own projects**.

1.1 MedicalNLP

There are several projects available on the use of NLP in the Icelandic healthcare system. The projects mostly consist of training language models for various tasks on health data from the database of the Capital Region Health Care (CRHC). So far, our projects have been about training language models on questions and answers to extracting clinical features from doctor notes. We have used these diagnostic features to train machine learning models to predict disease risk.

Now we focus on studying the ability of language models in chat mode on health data, a kind of assistant chatbot. It should be able to handle patient requests, e.g. to create prescription requests, go through their symptoms, find information about drugs and education about diseases. We have a huge collection of text data from the CRHC database, which is divided into general text data from healthcare professionals and chat data between healthcare professionals and patients.

The following are examples of projects that are suitable for master's and senior undergraduate students:

- Training more masked LMs to extract diagnostic features from health text. This task is similar to a Q/A task and also involves predicting whether a symptom is positive or negative if a description of it is found in the text. Models of interest are SpanBERT and ALBERT.
- The creation of IceGLUE - the language comprehension tests for large language models.
- Evaluate the basic functionality of various causal language models trained (e.g. Bloom, Sw3, Open Llama) and the ability to solve tasks on Icelandic health data.
- Creating an SBERT model for sentence similarity training.
- Classification of chat data.
- RLHF - Reinforcement Learning with Human Feedback - Implementation of pipelines and evaluation of the performance of different models on chat data.

1.2 Grammar checking

Develop a system which reads a text in some language and points to grammatical errors in it. In the case of Icelandic, you might search for feature agreement errors between subjects and verbs, agreement errors in noun phrases, between prepositions and the following noun phrases, etc.

Use tools like *IceNLP* or *Greynir* (or some equivalent tool for other languages than Icelandic) to perform tagging and parsing. One approach would be to implement a web interface (or a web service) for users to check text for grammatical errors.

1.3 Context-sensitive spell checking

Develop a program which points to potential context-sensitive spelling errors in some language. Use a corpus to train a (statistical) language model which can be used in the program for finding errors.

You could apply both word n-gram and PoS n-gram techniques for locating possible errors. However, make sure that your program only looks at words that are known, i.e. words that ordinary spell checkers will not point to as being incorrectly spelled.

1.4 Named Entity Recognition (NER)

NER is a subtask of information extraction that seeks to locate and classify atomic elements in text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

Develop a system based on hand-crafted rules or machine learning that performs NER for a language of your choice. You can use basic units like a PoS tagger and a parser as an aid.

1.5 Language Modeling

Develop a statistical language model or a neural language model trained on some corpus. Use some language model toolkit as an aid in the development. Compare and evaluate the models and build an GUI which allows a user to generate sentences based on the models.

1.6 Automatic thesaurus extraction

Develop a program which provides synonyms for the words of a given language. Use some available corpus required by the model that you choose to implement.

Much of the existing work on thesaurus extraction and word clustering is based on the observation that related terms will appear in similar contexts. Most systems extract co-occurrence and syntactic information from the words surrounding the target term, which is then converted into a vector-space representation of the contexts that each target term appears in.

1.7 Statistical parsing

The goal of this project is to develop a statistical parser for some language. For this you will need a parsed corpus (a treebank) for the given language for training the parsing model. The training consists of learning a probabilistic context-free grammar (PCFG), which is then used to assign the most likely parse tree to a sequence of words.

You can, for example, experiment with the Stanford parser – see `https://nlp.stanford.edu/software/lex-parser.shtml`.

1.8 Text summarisation

The goal of this project is to develop a system for single-document summarisation. The program takes a document as input and creates a summary that retains the most important points of the original document.

1.9 Intelligent Computer-Assisted Language Learning (ICALL)

ICALL is a relatively young field of interdisciplinary research exploring the integration of natural language processing in foreign language teaching.

Develop a system that helps students learn morphology/PoS tagging/shallow parsing in some language. The system might allow the user to input a sentence, analyse it and then give feedback based on an automatic analysis obtained using appropriate NLP components.

1.10 Intonation for Text-to-Speech

The goal is to get a speech synthesizer to sound more natural and intelligent. You would develop a program that inserts special intonation markers into text to be spoken by a Text-to-Speech system (TTS) based on various syntactic, semantic and pragmatic features. For example, you can use phrasal tones to distinguish between questions and statements, use pitch accents to highlight contrast between two options or to emphasize really interesting information (using the so called "information structure" of an utterance).

1.11 Question-Answering System

The goal is to get a system to give answers to questions about the contents of a text – all in natural language. For example, there could be a document describing the life and habitat of a certain animal. You would then perform semantic analysis on the text to populate a knowledge base with known facts (such as “cats eat mice” or eat(cats, mice)). A user can then type a question in natural language like “what do cats eat?” and the system would answer “mice”. It is fine to assume that the text is a basic text (maybe from a children’s book of knowledge).

1.12 Simple Dialog System

Similar to the Question-Answering System above, the goal of this project would be to have a natural language exchange with a system, but it would have to extend further than just one turn of interaction. Unlike the Q-A system, you can manually construct the required knowledge. The conversation should have a beginning (greetings), middle (maybe related to a task or a topic) and end (farewell).

1.13 Simple Spoken Q-A or Dialog System

Similar to the Simple Q-A and Dialog Systems above, except you would use both speech recognition and text-to-speech to support a spoken natural language exchange with the system. The emphasis here would be tying together all the different pieces of technology to make this possible rather than on making the conversation itself very deep. For special bonus points, incorporate intelligent intonation for the speech, which correctly emphasizes the most important information given by the system.

1.14 Simple Embodied Dialog System

Similar to the Simple Spoken Dialog System above, but incorporates an animated character that speaks. The conversation itself can be fairly simple (it can even follow a fixed branching dialog tree), but the goal is to use syntactic, semantic and pragmatic features in the text to automatically select appropriate nonverbal behavior (e.g. looking and pointing) for the character when speaking. To create an animation synchronized with speech, you can work with the virtual agents that were developed in the "Icelandic Language and Culture Training in Virtual Reykjavik" research project in the Socially Expressive Computing group[1].

1.15 Segmenting Icelandic Text

The goal is to automatically divide an Icelandic text into discourse segments (topics and sub-topics) and experiment with the so-called Attentional Stack model (a stack of topics essentially) for this purpose. This could be based on rules that detect various topic-change-related features in the text such as certain keywords (that belong to the category of “discourse markers”, such as “ok” og “semsagt”) and possible shifts in place, time, voice and etc.

[1] `http://secom.ru.is`

1.16 Label Noun Phrases with Information Status

The goal is to label all noun phrases in a text with the so-called *information status*, using a taxonomy proposed by Ellen Prince. In this taxonomy, entities referred to in an utterance may already be part of the conversation (evoked), inferred from what has been said (inferred) or new information (new). One benefit of knowing the information status of entities is to properly place special focus on new contributions, for example in speech or animation.

You would develop a program that automatically annotates this information status in an input text based on a dynamic *discourse model* (essentially a list of all entities mentioned so far). The most interesting part is to tackle possible inferred entities, such that a “door knob” would be considered inferred from a “door”.

2 Presentation of the project

The schedule for the selection and presentation of your project is the following:

- Friday, September 29th: Final date for the selection of projects (submit a reply to a post we will make on Piazza).
- Wednesday, October 16th: Status report in the form of a short oral presentation.
- Wednesday, November 8th: Final demonstration (exact time announced later).
- Friday, November 10th: Submit a research report (pdf file) and all code (source and executables), in a single .zip file or online repository such as GitHub, to the Final Project assignment that will be available on Canvas.

2.1 Format of the final report

Your final report should be a six page *research report*, containing sections on, for example, related work (background), implementation, evaluation and error analysis. We recommend following the **IMRaD** article structure.

The format of your final report (pdf) is a specific two column style. Latex and Word templates for this style are available on the **ACL GitHub** page. You may also use the **Overleaf template**.