Natural Language Processing: Final project

Reykjavik University – School of Computer Science

Instructors: Hrafn Loftsson and Hannes H. Vilhjálmsson

October 2015

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[1].

It is assumed that your system uses some of the techniques that have been (and will be) discussed in the course. **It is preferred that you work on this project in a group of two students**.

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

1.1 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 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] The teachers in the course can assist you in selecting the appropriate papers.

1.2 Machine translation

Develop a system which can translate texts from the source language S to the target language T, by using the so-called *shallow-transfer* method. Your program could perform shallow parsing on S and then translate each constituent, one by one.

Machine translation systems based on the transfer approach need to be able to map a word form in S to its lemma and then use the lemma for looking up the corresponding word in T (the target word then possibly needs to be inflected!). Note that if you want to develop a system for S=Icelandic then IceNLP includes a lemmatiser, in addition to a PoS tagger and a shallow parser.

Note that you could use Apertium (`http://www.apertium.org/`), the free/open-source machine translation platform, to develop your system.

1.3 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.

1.4 N-gram based text categorization

Develop a program which can classify text into one of five languages. You should derive a language model based on n-grams for each of the five languages and use the language models to classify unseen text.

You could, for example, implement a functionality similar to the one described in `http://www.let.rug.nl/~vannoord/TextCat/textcat.pdf`.

1.5 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.

1.6 Verb subcategorisation frames

Develop a program which automatically collects information about the subcategorisation frames of verbs in a given language. The program reads a corpus (or texts from the web), performs tagging and shallow parsing, and extracts information from the parsed text about the verbs and their objects. The output should be a list of verbs along with the subcategorisation frames.

The frames show the number of slots or arguments attached to a verb, i.e. does a verb demand one object, two objects, a prepositional phrase, etc. For Icelandic, we also need the information about the case of the object governed by a given verb.

1.7 An unknown word guesser

Develop a program which guesses the *tag profile* of an unknown word in some language. The tag profile is the set of possible PoS tags for the given word.

Your program uses a dictionary derived from some corpus. The dictionary contains word forms and the tag profile for each form.

When testing your program, a word is unknown if not found in the dictionary. The program performs some kind of morphological analysis on the unknown words and writes out the guessed tag profile for each unknown word. Note that the dictionary can be of help in the analysis, because it might contain information about words that are morphologically related to an unknow word.

1.8 Automatic thesaurus extraction

Develop a program which provides synonyms for the words of a given language. Use some available (very large) corpus and perform the necessary automatic annotation on it, needed by the method 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.

1.9 Experiments with an open source HMM tagger

The goal of the project is to experiment with the Hunpos tagger – a free and open source HMM PoS tagger (`http://mokk.bme.hu/resources/hunpos`).

You will be given an Icelandic PoS-tagged corpus to work with or you can use a PoS-tagged corpus in some other language. You train the tagger with various features settings in order to find out which settings is the most appropriate for the chosen language. In addition, you need to test these feature settings on training data of various sizes.

1.10 Experiments with a state-of-the-art, semi-supervised tagging algorithm

The goal of the project is to experiment with a state-of-the-art tagging algorithm, published at EACL in 2009 (`http://www.aclweb.org/anthology/E/E09/E09-1087.pdf`).

This method uses both manually tagged ("supervised") data and auto-tagged ("unsupervised") data. You will be given an Icelandic PoS-tagged corpus to work with or you can use a PoS-tagged corpus in some other language (except English). You train the tagger with various features settings in order to find out which settings is the most appropriate for the chosen language.

1.11 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 Berkely parser – see `http://nlp.cs.berkeley.edu/Software.shtml`.

1.12 Text summarisation

The goal of this project is to develop a computer program 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. You could, for example, implement the method describe in `https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf`.

1.13 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.

1.14 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). There are several free TTS systems out there that you can use, such as Festival [2]. We will also make the new Icelandic TTS system from Ivona[3] available for those interested in working with Icelandic speech.

1.15 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).

1.16 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

[2] `http://www.cstr.ed.ac.uk/projects/festival/`

[3] `https://www.ivona.com/`

should have a beginning (greetings), middle (maybe related to a task or a topic) and end (farewell).

1.17 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.

1.18 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 are present in the "Icelandic Language and Culture Training in Virtual Reykjavik" research project in the Socially Expressive Computing group[4].

1.19 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.

1.20 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),

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

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".

1.21 Generating an Image from a Natural Language Description

Write a program that can read in a description of a simple visual scene and from that description generate an actual image of that scene, for example using a 3D engine such as the Python-based Panda 3D[5]. You may constrain the language used in the description, but you must consider relative relationships and possible ambiguous references such as "next to the cube, there is a green sphere". The system may read the description interactively from a command line, in which case it could ask the user for clarifications when ambiguities arise. For example by asking the user "Do you mean the red cube or the green cube?".

2 Presentation of the project

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

- Friday, October 2^{nd}: Final date for the selection of projects (announced in an e-mail to the instructors).
- Friday, October 16^{th}: Status report in the form of an oral presentation (exact time announced later).
- Friday, November 6^{th}: Final demonstration (exact time announced later).
- Monday, November 9^{th}: Return of a research report (pdf file) and all code (source and executables).

[5] `http://panda3d.org`

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.

The format of your final report (pdf) is a specific two column style. Latex and Word templates for this style are available at `http://acl2014.org/CallforPapers.htm` (under heading "ACL 2014 Style files").