Samsacheck: A compound error checker for Swedish

Leo Marko Englund
Applied programming for linguists
Department of linguistics
Stockholm University
leo.markoenglund@gmail.com

Abstract

Samsacheck is a Python program designed to detect compound errors in Swedish. Compound errors here refers to incorrectly writing compounds as separate words. This is a common type of error in Swedish which is often missed by spell checkers as the compound parts are generally identical to correctly spelled independent words. Samsacheck features a customized part-of-speech tagger and a part-of-speech pattern detection module, aimed to catch frequent and clear types of compound errors. The tagger implements morphosyntactic feature extraction and an Averaged Perceptron machine-learning algorithm. It was trained and tested using the Stockholm-Umeå Corpus achieving an accuracy of around 95.8%. The compound detection in turn was preliminarily tested using a small set of development data, reaching an F-score of 0.47. This article describes in more detail the structure and functions of Samsacheck and discusses the difficulties in dealing with compound errors as well as possibilites for future development.

1 Introduction

Compounds are written as single words in swedish, incorrectly writing them as separate words however, is very common. The reasons for this error are various, separating the words might intuitively seem correct, writing conventions of other languages such as english might have an influence, or separation may be deliberate for visual reasons. Seldom does this error actually result in any serious miscommunication. Nevertheless, many people find it very disturbing, and unknowingly making these mistakes will lower the credibility of text.

Compound errors, as I will call them for simplicity's sake, are hard to catch by spell checkers. Not only are the isolated words often correctly spelled, the syntax may also be, or seem, correct. The most common mistakes follow some definable syntactical patterns: ”Jätte bra” (giant good, meaning great), ”Hamn arbetare” (dock worker), ”Spring pojke” (run boy, meaning errand boy). These patterns (noun adjective, noun noun, and others) are quite conspicuous, and the most easy to detect. Unfortunately, virtually any two words can form a compound.

The most notorious examples of compound errors are where the resulting sentence makes sense, but with a different meaning than intended, two examples being ”Sjuk sköterska” (sick nurse, meaning nurse), and ”Kyckling lever” (chicken is alive, meaning chicken liver). The first example is likely the result of very low Swedish proficiency, rather than a compounding error, as the reason to compound is semantically necessary to convey the proper meaning. The second example is more interesting, as the intended word is not 'lives' but 'liver' (both 'lever'), and if this is the case, there is syntactic motivation for compounding.

The aim of this project is to construct a Python program to detect compound errors. This program was given the name Samsacheck. Given the vast variety of compound errors, Samsacheck is made primarily to catch clearly conspicuous morphosyntactic patterns that match compounding patterns. This is done by machine learned customized part-of-speech tagging (POS-tagging), followed by a POS-tag pattern detection. Samsacheck also features an ambiguous tagging operation to catch cases such as ”kyckling lever”, and with more benefits discussed below. The project is experimental, but as a finished implementation it could be integrated in a spell checker for Swedish.

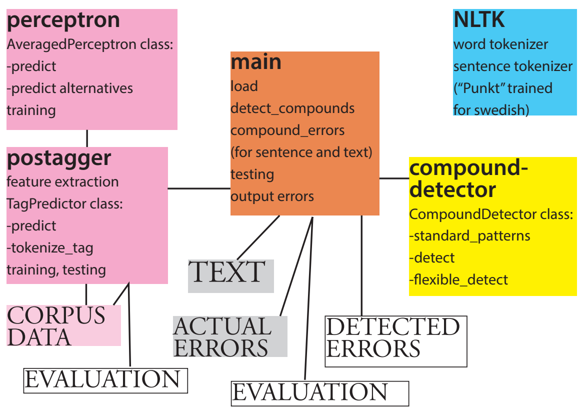
2 Data

The data used for machine learning for the POS-tagger is the Stockholm-Umeå Corpus (SUC) 3.0. SUC is a balanced collection of Swedish texts from the 1990's containing more than one million words. It is carefully annotated with POS-tags, and various morphological information. SUC was used to train as well as evaluate the POS-tagger for Samsacheck. However, rather than simply extracting POS-tags, certain specific forms, such as singular nominative indicative nouns (here called Base nouns), have their POS-tag replaced with a custom tag for that form. These forms can then be used in subsequent pattern recognition.

Another dataset is used for testing the actual compound detection. There is presently no dataset available with annotated compound errors. The data used for developing Samsacheck was derived from a recently published article from Dagens Nyheter (Gelin, 2016). This article was manipulated by separating compound words to create compound errors of various different types. An effort was made to have the compound errors differ in syntax, length and characters in the words, and only probable errors were included. However, being unnatural errors it is not certain how well they correspond to actual language data. The reason for this approach was to have a list of all errors to compare with the detections of the program.

3 Method

The operations of Samsacheck can be divided into two parts: training the POS-tagger, and running the detection program. A user will not need to run the first part, but instead use a saved model. An outline of the whole architecture is shown in the diagram below



For the first operation the SUC training data is read by the POS-tagger module. Only the word and the POS-tag is saved, although when certain combinations of POS- and other morphosyntactic tags are encountered the original SUC tag is replaced with a custom tag.

Each sentence of the data is passed to feature extraction, which is then performed for each word. All characters are converted to lower case, and the word, suffixes and to a lesser degree prefixes are extracted for the present word, preceding and following word to make a feature set. Two parallel list of feature sets and corresponding tags are used to train an Averaged Perceptron model.

The Averaged Perceptron model saves scores for each feature of the labels associated with it (in a dictionary of dictionaries). This is done by first predicting a label based on currently saved scores for each set of features. Then if the predicted label is wrong (or none) the score for the correct label for each feature in the set is increased by 1, and the score for the predicted label decreased by 1. If the prediction is correct, no action is undertaken.

The data is iterated more than once, for more accuracy. Too many iterations, however will result in overfitting to the training data. The number of iterations is possible to set at the time of training. Samsacheck has mainly been tested using a model trained with three iterations.

Two more collections of data are stored when training: the total scores for features and a timestamp for each feature. Also a counter is kept for the number of score updates. The totals for a feature consists of sums of the scores for classes associated with it at each update point. By using timestamps, the totals can be kept in line for all features although they are not all updated at the same time. Finally, the average scores derived from the totals is what is used for actual prediction/tagging, and the rest of the data need not be stored.

The second operation of Samsacheck handles raw text to detect compound errors. The text is tokenized using Natural Language Toolkit's (NLTK) sentence and word tokenizer. The tagging requires sentences to work correctly, as attributes from the previous and next word (if any) are added as features. The pattern recognition in the compound detector requires sentence boundaries as well.

The POS-tagger incorporating a previously trained Averaged Perceptron model is used to tag each word of each sentences. An ambiguous tagging, built into the Perceptron, is here the default choice. It returns all probable tags rather than just the top one. The threshold is defined in percent of the top scoring tag, and this threshold, or ”unambiguity level”, is possible to set by the user at each program run.

The sequence of tags (or tag sets) corresponding to each sentence are passed to a separate detector module performing pattern recognition. Patterns may be inputted, otherwise a standard set of patterns will be used. The most important of these is:

- Base noun – Noun (any form)
- Base noun – Adjective (any form)

Indices for detected compound errors are returned to the main module. Then in the end, output is given in the form of a text file listing the line number and two parts of separated compounds, in three columns.

4 Results and evaluation

Different tests were carried out for the POS-tagger and the final compound detection. For the tagger a standard machine learning testing was used, separating the data into a big training part, a small development test part and a small final test part. An already existing separation of SUC for these purposes was used. Testing is carried out by comparing the tags predicted with the actual tags, using the test data. Unambiguous tagging was used for these tests, as it is required for a clear result. The program outputs the results in amounts, accuracy and a report of all errors made, sorted by which tag was mistaken for which.

The accuracy of tagging according to these tests is around 95.8% for three iterations (the data is shuffled for each iteration while training, so the score may differ sligthly in either direction). The most frequent error was mistaking the custom group ”base nouns” with other nouns, and vice versa, and second most frequent was mistaking adjectives and adverbs or vice versa. Both of these errors will likely have less of an impact when using ambiguous tagging. Other than that a variety of infrequent or very rare errors were also found.

The accuracy of tagging was deemed high enough for the pattern recognition. However, the effectivity of this pattern recognition in turn needed testing. There is a serious problem here in the lack of extensive natural language data with annotated errors of the kind we are looking for. For a preliminary test data, an article with added compound errors (described earlier) were used, along with a list of those errors. The output of Samsacheck was compared with that list for evaluation.

Both accurate detections in relation to total detections (precision), and accurate detections in relation to total errors present (recall) were evaluated, as well as the harmonious mean of both (F-score). Results differed depending on the unambiguity level setting for tagging, the best being for 60% unambiguity: with 0.44 precision, 0.5 recall, and 0.47 F-score.

Upon human inspection of the output more was learned about the various reasons for errors. An important cause of non-detection is simply the lack of many possible compounding patterns. Samsacheck is still in an experimental stage and the number of patterns is not very great. The errors ”till ställning” (preposition noun) and ”själv ständig” (adjective adjective) exemplify patterns not (yet) covered by Samsacheck. Adding compounding patterns is possible but difficult, as many (maybe even most) patterns that could indicate a compound error could also be a completely normal word sequence. Making the pattern forms specific enough would help take care of this issue, but many patterns would likely still need to be omitted.

This brings us to the opposite problem, when correct word combinations resembling or actually matching an error pattern are identified as errors. Examples include ”gäng unga” and ”dussintal organisationer” where a pattern is matched as it should, but the word sequence is actually correct. Another type of case is where the tagging was incorrect or too ambiguous leading to an incorrect pattern detection, as in ”dryg månad”, where ”dryg” was tagged as a base noun instead or in addition to adjective. The first type of case especially gives a lot of important hints to grammatical rules Samsacheck is unaware of. This knowledge could lead to major improvements in further development.

Based on these tests, the patterns could be refined to both include more possible compound errors and to exclude more correct word combinations, possibly yielding much better results. The results in compound error detection of this approach would be interesting to compare with ones using an n-gram language model. It might be beneficial to implement both methods in combination.

5 Conclusions

This project involved creating a customized part-of-speech tagger utilizing the Stockholm-Umeå Corpus. In addition compound error patterns were defined and a pattern detection module made.

The part-of speech tagger implementing an Averaged Perceptron learning algorithm has a tagging accuracy of around 95.8%, for a separate testing part of the corpus. An ambiguous tagging is built in, giving multiple tags if their prediction score is close enough to the top scoring tag. The ambiguous tagging is beneficial to the following task of error detection, catching otherwise hard to catch sequences. An unambiguity level for tagging is possible to set on each program run, fitting the amount of output to the intended use (identifying more errors, but also non-errors, or listing only the clearest errors). The compound error detection has yet only been tested with a small amount of data. According to this test detection was given an F-score of 0.47 with 60% tag unambiguity. Further testing would be required to properly evaluate the performance.

However looking at the results, important lessons could be learnt about faults in the patterns definition. For example, sensitivity to grammatically correct sequences of noun-noun and noun-adjective would reduce incorrect error detection. There is much room for improvement while at the same time there are major difficulties in handling the wide variety of possible compound patterns, without identifying many correct forms as errors. It would be fruitful to compare results with this type of grammatical approach, to those achievable with an n-gram language model.

Samsacheck is freely available at https://github.com/leomarko/DaLi-project

References

Stockholm-Umeå Corpus 2012 Department of Linguistics, Stockholm University and Umeå University

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc. http://nltk.org/book

Martin Gelin. 2016. "Chelsea Clintons dilemma: att vara normal i USA:s minst normala familj" *Dagens Nyheter, DN.se*, 2016-10-21, 05.00.

Matthew Honnibal. 2013. "A Good Part of Speech Tagger in about 200 Lines of Python" *https://spacy.io/blog/part-of-speech-pos-tagger-in-python*, 2013-09-18, checked 2016-11-02.