Bielefeld University

Project Report

Contractions and Ambiguity

Grammar Based Disambiguation of English Apostrophe+S Contractions in Movie Scripts

Max Harder, 2919411
max.harder@uni-bielefeld.de

Module 23-TXT-BaCL5 Advanced Module
230028 Natural language processing:
Introduction to text analysis (S) (SoSe 2019)

Mr Nikolai Ilinykh
nikolai.ilinykh@uni-bielefeld.de

December 13, 2019

Contents

1 Introduction

Contractions occur frequently in spoken as well as in written English, especially in colloquial conversations and texts. When it comes to Natural Language Processing (NLP), they confront scholars and programmers with tricky problems; the meaning of contractions can be context dependent and not everything which looks like a contraction is one. Therefore, methods like part-of-speech (POS) tagging and an analysis of the underlying grammar structure are necessary to find contractions and determine their correct expansions. In this project, I focused on apostrophe+s contractions which are "the most frequent and the most ambiguous contractions" in the English language (Volk et al. 2011, p. 1). I developed an algorithm which identifies and expands seven different types of apostrophe+s contractions in movie scripts of ten different Quentin Tarantino movies. In doing so, I made use of a categorisation of contractions compiled by Volk and Sennrich (2011) in their paper on "Disambiguation of English Contractions for Machine Translation of TV Subtitles". In this paper, the authors also provide grammatical patterns which can be used to identify the correct type of contraction. To make use of these grammatical patterns, my algorithm employs POS tagging, phrase chunking, and named entity recognition (NER), but ignores automatically generated POS tags for apostrophe+s tokens. This provides an alternative to the unsatisfying results of the simple dictionary based *contractions* module and to POS taggers that only use limited context windows (ibid., p. 5); beyond that, *TreeTagger*, the POS tagger used in this project, does not even have tags for apostrophe+s as possesive marker or pronoun (ibid., p. 5). All in all, the result of the algorithm suggested in this project seem to be more reliable than those of common POS taggers.

2 Data

The central algorithm of this project is included in a Jupyter Notebook called `contractions_project.ipynb`. The data for this project comes from ten text files, each containing a different Quentin Tarantino movie script which was freely accessible on one of various internet platforms. The folder `data`, a subfolder of the main project folder, contains the following files used by the algorithm:

1. quentin-tarantino_django-unchained.txt
2. quentin-tarantino_four-rooms.txt
3. quentin-tarantino_from-dusk-till-dawn.txt
4. quentin-tarantino_inglorious-basterds.txt
5. quentin-tarantino_jackie-brown.txt
6. quentin-tarantino_kill-bill.txt
7. quentin-tarantino_natural-born-killers.txt
8. quentin-tarantino_pulp-fiction.txt
9. quentin-tarantino_reservoir-dogs.txt
10. quentin-tarantino_true-romance.txt

After running the preprocessing function included in the Jupyter Notebook, an additional 11^{th} file is created in the `data` folder:

11. PREPROC_scripts.txt

A second folder called `output` is created after running the main functions of the Jupyter Notebook. This folder contains the following files:

1. MISSING_scripts.txt
2. OUTPUT_scripts.txt

3 Preprocessing

A challenge in preprocessing the data was that each movie script is formatted in a different way; while some scripts contain a lot of white space and line breaks, just like a .pdf file, others have a single paragraph in each line. In a first step, the preprocessing algorithm finds all paragraphs of each file to ensure that it processes full sentences. In doing so, it scans every line of each text file and combines it with the previous line if it belongs to the same paragraph. This is possible since in almost all cases paragraphs are separated by an empty line. In a next step, the algorithm normalises each sentence in a paragraph and then checks it for apostrophe+s contractions. All sentences which contain at least one such contraction are transferred as a single line to `PREPROC_scripts.txt`.

4 Processing

The grammatical patterns used in the processing part of the algorithm are taken from Volk and Sennrich (2011) and are slightly adapted to fit into the frame of this project. In their conference paper, the authors distinguish seven different types of contractions which I specify in the following list, indicating the corresponding POS tag after the em dash[1] and the underlying grammatical pattern as bullet points:

1. the **possesive marker**—POS
    - following a name or a noun
    - the beginning of a noun phrase in the article position
2. the **copula 'is'/'was'**—VBZ (copular)
    - in front of a noun phrase or an adjective (phrase)
    - *no distinction between present tense 'is' and the past tense 'was'*
3. the **auxiliary 'is'/'was'**—VBZ (auxiliary)
    - a following verb in present participle form
    - perhaps a 'not' or an adverb intervening
    - *no distinction between present tense 'is' and the past tense 'was'*
4. the **auxiliary 'has'**—VHZ
    - in front of a past participle verb form
    - perhaps a 'not' or an adverb intervening
5. the **auxiliary 'does'**—VDZ
    - a question and a verb that is neither a present nor a past participle
6. the **pronoun 'us'**—PP (us)
    - after 'let'
7. the **plural marker**—N/A (plural marker)

[1] The tags are found in the output file to specify the preceding expansion. They are, if existent, in line with the POS tags of the *Penn Treebank Project*, which I also used for the *TreeTagger* in this project.

- following a number and occurring at the end of the sentence
- following a number that is following a plural indicator like ‘all’ or ‘many’ (Volk et al. 2011, p. 4 f)

In the processing function, I have used a context window with a maximum size of two to check each contractions for its type. A context window with a size of two causes problems when the algorithm cannot check the surrounding two words, i.e. the contraction is the last word or it precedes the last word of the sentence, or it is the first word or it follows the first word, respectively. This, however, is only a pattern specific problem, but not a problem in general because most of the patterns do not use context windows with a size of more than one. The most trivial case, for example, is the pronoun ‘us’; the only condition is that the preceding word is “let”. The most important information for most conditions comes from the POS tags. To identify the copula ‘is’/‘was’, however, noun chunks are used to check whether the apostroph+s contraction is in front of a noun phrase. Additionally, NER is used to check whether the preceding word is a named entity; if it is, the contraction is likely to be a possesive marker. If none of the algorithm’s if-conditions is fulfilled, the patterns were not strong enough to identify the correct type of the contraction. In this case, the contraction sentence is not modified with a specified contraction and transferred to the output file `OUTPUT_scripts.txt`, but transferred to an additional file `MISSING_scripts.txt` in the `output` folder which collects all unindentified contractions, indicating the POS tags of the apostrophe+s and its preceding and following token, the three tokens themselves, as well as the full sentence.

5 Results

Keeping the simplicity of the approach in mind, the output of the algorithm is surprisingly correct. Even without a substantial evaluation, which would exceed the set time frame of the project, it becomes apparent that almost all occurrences of apostrophe+s contractions in the output file are attributed to the correct contraction type. The main problem seems to be connected to the auxiliary ‘has’ (VHZ) which is sometimes misattributed in auxiliary ‘is/was’ (VBZ (auxiliary))

cases, e.g. “marsellus brings up his weapon and fires , but he (’has’, ’VHZ’) so hurt , shaky and dazed that his arm goes wild .” (OUTPUT_scripts.txt, l. 233) Moreover, it should be noticed that the algorithm did not identify a single auxiliary ‘does’ contraction. The reason for this might be that its occurrences are so rare – even Volk and Sennrich ignored this case (Volk et al. 2011, p. 5) – or that the grammatical pattern is not sufficient. All in all, the algorithm identified contractions in 3,637 of 4,881 sentences, i.e. it was not able to identify the type of contraction in 1,234 sentences. This reduces to an identification rate of about 75% on the basis of sentences, which can contain multiple contractions. To further improve the results in future applications, the alorithm can be enriched with information gained through a POS tagger. This would create the opportunity to simplify the algorithm in simple cases, to increase the identification rate, and maybe to reduce the misattribution of the auxiliary ‘has’. A more difficult task will be to disambiguate apostrophe+s contractions which can mean present tense ‘is’ or past tense ‘was’, a problem which a left out intentionally.

References

Volk, M and R Sennrich (2011). "Disambiguation of English contractions for machine translation of TV subtitles". In: *NODALIDA 2011*. Nordic Conference of Computational Linguistics, Riga, 11 May 2011 - 13 May 2011.

Written Insurance

I hereby certify that I have prepared this written project report independently. All passages which are taken from the wording or the meaning of other works (including electronic sources) have been clearly marked in each individual case with precise indication of the source.

Bielefeld, December 13, 2019

_____

(Max Harder)