# Towards Automatic Syntactic Simplification: A Case Study on Identifying Tree Transformations

## 1 Introduction

Written documents tend to be composed of rather complex sentences. For example, the average length of a sentence in the Penn Treebank is about 18 words. There are many situations in which such complex sentences are difficult to process, either by humans or by computers. In such cases, sentence simplification methods can be used to generate simpler sentences. Simplification may replace complex words by more basic words, replace complex grammatical constructions by simpler ones, make implicit assumptions explicit, or apply any combination of those. The range of targeted phenomena is generally determined by the consumer of the simplified texts. If young readers are targeted, all strategies would apply while a reading aid for aphasia patients would focus on the specific phenomena, such as passive sentences, that are difficult for such patients to process.

In our project, the goal is a trainable natural understanding component that can convert text to a semantic representation based on event semantics [9]. Since learning a conversion to logical form for unrestricted sentences is a rather daunting task, we simplify sentences before we perform the semantic analysis. In our scenario, syntactic simplifications are the most important type of simplification. Our intention is to develop a method for simplifying any given syntactic phenomenon based on a small set of sentences that exhibit the intended phenomenon, and for which we know the intended simplified form. This method will be based on tree transformation patterns that are extracted from the regular/simplified sentence pairs. The current paper presents a case study on identifying such patterns for the simplification of passive sentences into active ones.[1] To date, this problem has been approached by manually written transformation rules [1, 2, 5, 10]. For this reason, our first approach was to use a tree transducer to automate the actual tree modifications. However, this proved ineffective since the rules had to be extremely specific. Our novel approach relies on the automatic identification of chunks[2] that are moved to a new position and chunks that are deleted or added. Aggregating over the set of sentences allows us to extract transformation patterns. For this paper, we compare the rule-based and chunk-based approach in terms of coverage and accuracy.

[1]The final paper will present results for reported speech, omitted here due to space limitations.

[2]Note that we define chunks as sequences of words, as opposed to definitions used for chunk parsing.

## 2 Method

**Data** We collected a corpus of passive sentences and their simplified, active forms from a sentence-aligned parallel corpus of regular and simple[3] English Wikipedia and from the SynCOIN data set [3], where simplified versions were provided manually. We currently concentrate on sentences that contain the target phenomenon without other sources of complication. The parallel sentences were parsed using the Berkeley parser with a model trained on the Penn Treebank [7].

**Rule-Based Approach** Our rule-based approach relies on a tree transduction package, the Keen Utility for Rewriting Trees (KURT) [8]. The rules are written in YAML[4] notation and specify the syntactic structures (sub-trees) for rule application, to which the transformation is to apply, as well as the resulting transformed subtrees. We will provide an example of a rule in the final paper. KURT rules are generalizable in the sense that the variables on the left-hand side can match arbitrary subtrees. But when we need to manipulate specific elements of a subtree on the right-hand side, the internal character of the subtree needs to be stated more explicitly, in terms of numbers and potentially types of daughters. Since our sentences can have a wide variety of modifications, this leads to an ever-increasing number of rules, even in a restricted task domain such as transforming passive sentences to active.

**Chunk-Based Approach** Our novel method relies on aligned chunks between the regular and simplified sentences. We identify chunks (i.e., sequences of words that remain adjacent, but not necessarily in the same position) between the sentences by considering the sentences as sequences in a classical longest common subsequence (LCS) problem. We then apply the LCS algorithm [4] iteratively while relaxing the constraint that the items in the sequences (i.e., the words) be strictly equal; rather, we consider them a match if their normalized edit distance [6] falls within an experimentally determined threshold. The latter is necessary to account for changes in the inflected forms, for example from the passive verb form to the active one. The common subsequences, in other words the consecutive sequences of words shared by the two sentences, are then considered chunks. Non-aligned words are not considered further, which translates into a deletion within the simplified sentence.

Once we have the chunks and their linear positions within each sentence, we identify the chunk that serves as an *anchor*. An anchor is a fixed chunk around which the other chunks may be arranged. Then, we perform the subsequent re-arrangements.

We illustrate the method by means of an example. Consider the regular, passive sentence “Major scientific breakthroughs were achieved by French scientists in the

[3]`http://simple.wikipedia.org/wiki/Main_Page`
[4]`http://www.yaml.org/`

18th century .” It is paired with the corresponding simplified sentence “French scientists achieved major scientific breakthroughs in the 18th century .” After iteratively applying the LCS algorithm, we obtain the following chunks:

1. Major scientific breakthroughs
2. achieved
3. French scientists
4. in the 18th century .

Parsing the two sentences with the Berkeley parser, using a model trained on the Penn Treebank, we receive the following trees:

```
( (S
    (NP (JJ Major) (JJ scientific) (NNS breakthroughs))
    (VP
      (VBD were)
      (VP
        (VBN achieved)
        (PP (IN by) (NP (JJ French) (NNS scientists)))
        (PP (IN in) (NP (DT the) (JJ 18th) (NN century)))))
    (. .)))

( (S
    (NP (NP (NNP French) (NNS scientists)))
    (VP
      (VBD achieved)
      (NP (JJ major) (JJ scientific) (NNS breakthroughs))
      (PP (IN in) (NP (DT the) (JJ 18th) (NN century))))
    (. .)))
```

We then find the minimal subtree in the regular parsed tree which spans over all the leaves of each candidate chunk. We note that the minimal subtree for candidate chunk 4. is actually the full sentence, since the period is dominated by `S` instead of being grouped in the `PP` constituent. The period is consequently split off into its own chunk. Other non-constituent candidates are handled similarly.

1. `(NP (JJ Major) (JJ scientific) (NNS breakthroughs))`
2. `(VBN achieved)`
3. `(NP (JJ French) (NNS scientists))`
4. `(PP (IN in) (NP (DT the) (JJ 18th) (NN century)))`
5. `(. .)`

Once each chunk is associated with a minimal spanning subtree, we put them through a filter to see which chunk serves as an *anchor*, around which the other chunks are redistributed. That is, we need to determine which chunk is the most stable with respect to its position in the parse tree and position in the sentence. Our assumption for the passive-to-active conversion is that this should be the verb. But it may be another chunk for other conversions. Since "achieved" maintains the same mother node and position from the regular to simplified trees, it is chosen as the anchor.

The 4 remaining chunks and their associated minimal spanning subtrees are probed to determine which tree properties are associated with the rearrangements from the complex to simple subtrees. Since the candidate chunk "French scientists" is dominated by a *by*-phrase, whose mother is a `VP`, it is assigned to move to the left of the anchor. "Major scientific breakthroughs" has the tree root, `S`, for a mother and originally occurs to the left of the anchor, thus it is moved to the right. The adjunct and the punctuation remain *in situ*. The resulting sentence matches the simplified sentence: "French scientists achieved major scientific breakthroughs in the 18th century ." This describes our intuition that the original subject moves into an object position while the `NP` of the "by"-`PP` becomes the subject. For the future, we are planning to implement a machine learning approach that identifies such patterns across a small training set of sentence pairs using a set of tree substructure templates.

## 3 Evaluation

In this section, we present an evaluation of the approaches in terms of coverage and accuracy. A *correct simplification* has the same linear ordering of the leaves as the gold simplified sentence. I.e., we currently do not performing any morphological adaptation of the word forms.

### 3.1 KURT

As mentioned previously, KURT rules must be specific to capture different features of the input. In table 1, we can see how many additional KURT rules are needed as a function of the number of sentences covered. We used a development set of 62 sentences for this evaluation. A base of 10 rules is needed for the first 10 sentences in the development set, followed by an average of 4 additional rules needed for each 10 additional sentences covered. It is unclear how many more rules need to be added to cover a reasonable set of sentences, but we assume, there is a wide range of modifications in the tree structure that we have not seen in our small data sets. After developing the rules on the development set, we tested them on a test corpus of 32 sentences. The results are shown in table 2. The KURT rules achieved an accuracy of 65.62%, i.e., they correctly simplified 21 out of 32 sentences. The incorrect sentences were assigned reasonable parses but they presented the KURT

| # of sents | Additional rules | Total rules |
|---|---|---|
| 10 | 10 | 10 |
| 20 | 5 | 15 |
| 30 | 3 | 18 |
| 40 | 3 | 21 |
| 50 | 4 | 25 |
| 62 | 5 | 30 |

Table 1: Additional KURT rules as a function of number of sentences covered.

| KURT | Chunk |
|---|---|
| 65.62% | 93.75% |

Table 2: Accuracy of both methods on the test set.

rules with unfamiliar syntactic structures, especially in the main branches of the trees where the transforms take place.

### 3.2 Chunk-based

The chunk-based approach was aggregated over the 62 sentence development corpus, where transformation patterns were extracted from the candidate chunks. Observations made over the data gave insight into the properties that can serve as reliable indicators of which chunk should move left of the anchor, right of the anchor, or should stay *in situ*. After integrating this information and achieving complete coverage of the development set, we tested against the 32 sentence test corpus. The results of evaluating the accuracy of our approach are shown in table 2. We achieved an accuracy of 93.75%, or 30 out of 32 sentences correctly simplified. Both incorrectly simplified sentences had incorrect chunks, as in the following example.

**regular:** *In the 30s, 70s and 90s the* Underground was bombed many times by the IRA.

**simplified:** *In the 30s, 70s and 90s the* IRA bombed the Underground many times.

## 4 Conclusion

In this paper, we have started to develop an automatic method for simplifying sentences. Our experiments show that KURT requires a large set of very specific rules while the chunk-based approach is more general and more accurate across unseen examples. For the future, we are planning to extend the work to identify the patterns of rearranging chunks automatically by detecting regularities across sentence pairs.

## References

[1] John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. Simplifying English text for language impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Bergen, Norway, 1999.

[2] Iustin Dornescu, Richard Evans, and Constantin Orašan. A tagging approach to identify complex constituents for text simplification. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 221–229, Hissar, Bulgaria, 2013.

[3] Jacob Graham, Jeffrey Rimland, and David Hall. A COIN-inspired synthetic dataset for qualitative evaluation of hard and soft fusion systems. In *Proceeding 14th International Conference on Information Fusion*, Chicago, IL, 2011.

[4] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, June 1975.

[5] Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. Text simplification for reading assistance: A project note. In *Proceedings of The Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications*, Sapporo, Japan, 2003. ACL.

[6] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707, 1966.

[7] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[8] Alex Rudnick. Tree transducers, machine translation, and cross-language divergences. `http://arxiv.org/abs/1203.6136`, 2012.

[9] Roser Saurí, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. TimeML annotation guidelines. `http://www.timeml.org/site/publications/specs.html`, 2006.

[10] Advaith Siddhartan. An architecture for a text simplification system. In *Proceedings of the Language Engineering Conference 2002 (LEC 2002)*, pages 64–71, Hyderabad, India, 2002.