Going Dutch: Creating SimpleNLG-NL

Ruud de Jong
Human Media Interaction
University of Twente
Enschede, The Netherlands
r.f.dejong@utwente.nl

Mariët Theune
Human Media Interaction
University of Twente
Enschede, The Netherlands
m.theune@utwente.nl

Abstract

This paper presents SimpleNLG-NL, an adaptation of the SimpleNLG surface realisation engine for the Dutch language. It describes a novel method for determining and testing the grammatical constructions to be implemented, using target sentences sampled from a treebank.

1 Introduction

SimpleNLG is a Java-based surface realisation library aimed at practical applications (Gatt and Reiter, 2009). It is meant to be simple to use, and in an architecture redesign in version 4 of the software it was also made easy for developers to alter its code. Over the years, SimpleNLG has been adapted for several languages other than English, such as German (Bollmann, 2011), Brazilian-Portuguese (De Oliveira and Sripada, 2014) and Italian (Mazzei et al., 2016).

In this paper we present a new version of SimpleNLG for surface realisation in Dutch, called SimpleNLG-NL. Like most adaptations for other languages, it is based on SimpleNLG-EnFr (Vaudry and Lapalme, 2013): a bilingual version of SimpleNLG that supports both English and French, based on SimpleNLG version 4.2. The architecture of SimpleNLG-EnFr was split into language independent and language dependent parts, making it relatively easy to add new languages.

As Dutch is closely related to German, SimpleNLG for German (Bollmann, 2011) might seem a more obvious starting point for SimpleNLG-NL. However, SimpleNLG for German is based on the differently structured version 3 of SimpleNLG, which made it unsuitable to build on.

This paper is structured as follows. In Section 2 we describe the method we used for developing SimpleNLG-NL, followed in Section 3 by an overview of the main characteristics of Dutch and how we implemented them. In Section 4 we present the current coverage of SimpleNLG-NL over a set of test sentences. We end with conclusions and directions for future work.

2 Method

Instead of following the structure of e.g., a grammar reference book for adapting the rules of SimpleNLG, we developed SimpleNLG-NL using target sentences sampled from a dependency treebank for Dutch. In SimpleNLG-IT (Mazzei et al., 2016) sentences from a dependency treebank were used as well, but only for evaluation purposes. For SimpleNLG-NL, we expanded their use to the development phase, using them in an iterative generate-evaluate-revise process.

For each sentence, the SimpleNLG input was written manually and the resulting realisation was compared with the target sentence. Differences between the realisation and the target sentence were analysed. Based on this, the relevant grammar rules were adapted for Dutch and missing lexicon entries were added. This process was repeated for each sentence, increasing the size of the covered grammar subset with each iteration.

The generate-evaluate-revise cycle was carried out in four rounds.

First round: In the first round we tried to reproduce 12 target sentences of increasingly higher word count. We assumed that increasing the word count would also increase the grammatical complexity of the sentence.

Second round: In the second round we applied unit tests to 37 short sentences that were manually written to test one feature of SimpleNLG-NL, or a combination of very few features. Features that were tested included basic verb inflec-

Proceedings of The 11th International Natural Language Generation Conference, pages 73–78,
Tilburg, The Netherlands, November 5-8, 2018. ©2018 Association for Computational Linguistics

tion, both regular and irregular, as well as adjectives and their inflection (10 sentences), morphology of different verb groups in multiple tenses (12 sentences) and syntax of negated sentences (5 sentences). Finally, 10 sentences were used to test interrogative sentence types.

Third round: In the third round we included two more sets of target sentences from the Dutch treebank. The first set consisted of 11 medium-length sentences (7-13 words). The second set consisted of 10 long sentences (14-20 words).

Fourth round: In the final round, 16 more unit tests were carried out. These were aimed at testing combinations of tenses, voices and verb form (perfect or simple). They were based on the same input sentence, which contained a subject, a verb and a direct object.

The test sentences for Rounds 1 and 3 were randomly selected from the Dutch Wikipedia corpus (100,000 sentences) available in Dact,[1] a viewer for Alpino corpora. Alpino is a dependency parser for Dutch (Bouma et al., 2001).[2] After a sentence was randomly picked, based on the word count needed for the current round, it was tested for two requirements: the sentence had to be grammatically correct and it should not contain direct speech. SimpleNLG does not support properly embedding direct speech in a sentence and neither would SimpleNLG-NL.

For each sentence that was selected, the input code for generation had to be written. We did this based on its dependency tree from the treebank, as generated by the Alpino parser. As the input for SimpleNLG is structured similar to a dependency tree, the Alpino trees could fairly easily be converted into input code. Similar to the conversion rules used in the evaluation of SimpleNLG-IT, when converting the dependency trees to SimpleNLG we kept the input isomorphic to the tree in terms of subject, object etc. We did not use canned text in the input (except for names and fixed multi-word expressions), and we did not provide information about word order or punctuation.

We started the creation of SimpleNLG-NL by cloning the French parts of SimpleNLG-EnFr. French was chosen because its features seemed more related to Dutch than the English features, in particular with respect to the more complex morphology of French and Dutch compared to English. Therefore, the first realisation results used French grammar rules. For the lexicon, we translated the closed classes from the French lexicon (106 entries). The open part of the lexicon started out empty; missing entries (for irregular word forms) were added as we went along. A full lexicon was created later, as described in Section 3.5.

3 Adaptations for Dutch

The main changes we made to SimpleNLG for surface realisation in Dutch are described in this section. As our main references for Dutch grammar and morphology we used the online linguistic resources Taalportaal[3] and e-ANS.[4]

3.1 Nouns

In Dutch, pluralisation of nouns almost always consists of adding either *-en* or *-s* as a suffix to the singular form. Which suffix to use is determined by the stress of the noun's last syllable: use *-en* when stressed; use *-s* when unstressed. However, since we do not have information on stress, and there are many exceptions to this rule, SimpleNLG-NL instead uses a set of word endings to determine when to use the *-s* suffix.[5] Other exceptions can be added to the lexicon. Dutch also has compound nouns. Compounds are currently treated as regular nouns. Because of that, pluralisation may result in an incorrect form if the compound is not in the lexicon and does not have one of the predetermined word endings described earlier. When adding plural suffixes, other morphological rules may apply to ensure the correct spelling of the surface form. Specifically, sometimes vowels need to be removed or consonants added to the noun stem.

3.2 Verbs

The following tenses have been implemented in SimpleNLG-NL: present simple, past simple, present perfect, past perfect, future and conditional. When inflecting verbs, SimpleNLG-NL first checks the lexicon for irregular verb forms. If none are found, it uses the rules for regular verbs. Inflections are based on the stem of the verb. For

[1] http://rug-compling.github.io/dact/

[2] http://www.let.rug.nl/~vannoord/alp/Alpino/

[3] http://www.taalportaal.org/

[4] http://ans.ruhosting.nl/e-ans/

[5] http://ans.ruhosting.nl/e-ans/03/05/03/body.html

example, past tense inflection involves adding the suffix *-te* if the stem ends in an unvoiced consonant; in all other cases *-de* is added. Similar to nouns, in some cases the spelling of the verb stem needs to be changed.

Auxiliary verbs have to be added in the future tense (*zullen* 'will') and the conditional tense (*zouden* 'would'). They are placed before the verb. The perfect form also requires one of two auxiliary verbs (*zijn* 'be' or *hebben* 'have'), which can be specified in the lexicon. Lastly, passive sentences require *zijn* 'be' to be added. When these features are combined, such as in a passive conditional perfect sentence, in the current version of the system this results in an incorrect order of auxiliary verbs. In some cases, one or more auxiliary verbs have to be placed after the main verb, but the system currently does not do this.

Separable Complex Verbs. In Dutch, a special group of verbs is that of the so-called *Separable Complex Verbs* (SCVs). An SCV is a verb that consists of a main verb and a prepended *preverb* (Booij and Audring, 2018). This preverb can be any word, but is often a preposition.

In the past and present simple tenses in main clauses, SCVs are split into their preverb and main verb, and their order is reversed. The main verb is inflected as it would if it were on its own. For example: *toekennen* in the third person present becomes *hij kent toe* ("he assigns"). The position of the preverb in the main clause is flexible: direct objects, indirect objects, prepositional phrases and even entire subclauses can be placed between the main verb and its preverb. In SimpleNLG-NL, we decided to position the preverb at the end of the sentence by default. In the perfect tenses and in subordinate clauses, the preverb attaches to the main verb. The main verb is inflected normally, and the preverb is prefixed to it after inflection. This results in, for example, *hij heeft toegekend* ("he has assigned") and *dat hij toekent* ("that he assigns").

The input for an SCV can be in either of two forms: *toe|kennen* or *toekennen*. The first input splits the verb *kennen* from the preverb *toe*. SimpleNLG-NL will then look for *kennen* in the lexicon and inflect it appropriately (either from lexicon data or regular verb rules). This is similar to how SimpleNLG for German deals with such verbs (Bollmann, 2011). In the second case, SimpleNLG-NL checks if the verb is marked as an SCV in the lexicon using the `<preverb></preverb>` field. If it is not, SimpleNLG-NL tries to detect if the verb is an SCV based on a list of common SCV prefixes: *bij, in, na, uit, op, af, mee, tegen, tussen, terug, toe*. However, not all SCVs can be caught this way, as several verbs prefixed by a preposition are not SVCs, but look exactly like them with the difference being the stress. For example, *doorboren* (regular verb: 'pierce'; SVC: 'continue drilling') is only an SVC if the stress is on *door*.

3.3 Adjectives

Dutch adjectives can be used predicatively and attributively, with only the latter being supported by SimpleNLG and SimpleNLG-NL. Depending on the number and gender properties of the noun phrase, the adjective requires the suffix *-e*. Similar to nouns and verbs, in some cases the spelling of the stem needs to be changed.

Comparatives and superlatives are created with a suffix (*-er* or *-st*). In some cases, the adverbs *meer* ("more") or *meest* ("most") are used instead. In all cases, the adjective can be appended with the earlier mentioned *-e*. Comparative and superlative forms can be overwritten in the lexicon using the `<comparative></comparative>` and `<superlative></superlative>` fields.

3.4 Word order

SimpleNLG-NL uses the subject-verb-object order for main clauses and subject-object-verb for relative clauses and interrogative sentences. Exceptions are made for SCVs, as described in Section 3.2. The order of other constituents, specifically modifiers, can vary depending on many factors. Currently, SimpleNLG-NL allows for manipulating word order by specifying modifiers as 'premodifiers' or 'postmodifiers' in the input; otherwise, a default (and not always correct) word order is chosen.

3.5 Lexicon

A lexicon was created by parsing the Dutch pages of Wiktionary[6]. The content is licensed under the CC BY-SA 3.0 license[7], which makes it suitable for release with SimpleNLG-NL. This resulted in a lexicon containing 79437 entries (nouns, verbs, adverbs, adjectives and prepositions), including

[6] https://www.wiktionary.org/

[7] https://creativecommons.org/licenses/by-sa/3.0/deed

Sentence set	Sentences	Exact matches		Accepted as correct	
Round 1	12	8	66.7%	11	91.7%
Round 2	37	37	100.0%	37	100.0%
Round 3 (medium)	11	9	81.8%	9	81.8%
Round 3 (long)	10	5	50.0%	7	70.0%
Round 4	16	10	62.5%	10	62.5%
Total	86	69	80.2%	74	86.0%

Table 1: The final coverage of SimpleNLG-NL after development and testing. Generated sentences were "accepted as correct" if they met the criteria described in Section 4.

the closed part described in Section 2 and the entries added during the development rounds.

To accommodate making a choice in the trade-off between larger lexicons that take longer execution time and smaller lexicons that may miss required entries (cf. (De Oliveira and Sripada, 2014)), two smaller lexicons were generated based on subsets of the larger lexicon. The subsets were determined by matching the entries with word forms from a word frequency list based on OpenSubtitles.[8] Words in the frequency list can have multiple corresponding lexicon entries. The smallest lexicon, based on the top 1000 most frequent words, contains 3386 entries. The top 10,000 words result in 8600 entries. The full lexicon is over 10 MB, while the medium one is just over 1MB and the smallest is half of that. The choice of lexicon is based on scope and performance requirements. By default, SimpleNLG-NL uses the medium lexicon.

4 Evaluation

To determine the coverage of SimpleNLG-NL, each sentence generated in one of the four rounds described in Section 2 was judged on correctness. Since the number of sentences to be evaluated was small, using automated evaluation metrics such as BLEU (Papineni et al., 2002) would not have made much sense; moreover, these would not take into account that word order in Dutch is relatively free. Therefore we chose to manually evaluate the sentences.

We considered a sentence generated by SimpleNLG-NL to be generated "correctly" if the output met at least one of the following criteria:

- The output matched the target sentence exactly, including punctuation; or
- The output only differed from the target in terms of punctuation (commas and quotation marks), with no change in meaning; or
- The output differed from the target in terms of word order, but without making the sentence unwellformed or causing a change in meaning.

The criteria are ordered by inclusiveness, with the first being the preferred outcome ("exact match"). The final coverage by SimpleNLG-NL of the test sentences according to these criteria, after all four rounds of generate-evaluate-revise, is shown in Table 1.

Results round 1: Out of 12 sentences, 11 were generated correctly (91.7%). The result counting only exact matches is 8 out of 12 (66.7%). Of the three accepted mismatches, one missed some non-mandatory commas, and two had acceptable differences in word order from their target sentences. (One lacked topicalisation, which is currently unsupported, and the other placed the past participle at the end of the sentence, a merely stylistic difference.) SimpleNLG-NL could not reproduce the longest sentence from Round 1 (26 words). This was due to several problems. First, SimpleNLG cannot handle clauses without verbs, in this case an enumeration ("tasks such as X, Y and Z"). Second, the sentence contained a verb cluster as well as an attributively used infinitive, neither of which SimpleNLG-NL could handle.

Results round 2: In Round 2, all 37 short test sentences were generated correctly, as exact matches (100%).

Results round 3: Of the 11 medium-length sentences, 9 were generated as exact matches (81.8%) and the same number were accepted as correct. The two incorrectly generated sentences both had problems with modifier ordering. Of the 10 long

[8] https://github.com/hermitdave/FrequencyWords/

sentences, 7 were generated correctly (70.0%). This includes two sentences that did not match exactly. One accepted mismatch added an unnecessary (but acceptable) comma, the other positioned the preverb of an SCV at the end of the sentence. While that position is acceptable, it can be stylistically preferable to reduce the distance between the main verb and the preverb. However, SimpleNLG-NL does not yet support such a stylistic mechanism. The problems with the three incorrectly generated sentences involved incorrect ordering of modifiers and a verb cluster (*te gaan wonen*, lit. "to go live"), and lack of support for main clauses connected by a semi-colon.

Results round 4: Of the 16 varieties of the same sentence, 10 were generated as exact matches. There were no mismatches accepted as correct. The incorrect sentences all had an incorrect word order. Active sentences in the future perfect and the conditional tenses incorrectly positioned the auxiliary verb before the object. In passive sentences in the perfect form, the order of the verb and the two or three auxiliary verbs was incorrect (e.g., *zal zijn geweest gegooid* should be *zal gegooid zijn geweest* "will have been thrown").

Overall results: In total, 74 out of 86 test sentences (86.0%) were generated correctly. Of these, 69 (80.2%) are exact matches. If we only look at the 33 treebank sentences from Rounds 1 and 3, then 28 (84.8%) were generated correctly, with 22 (66,7%) exact matches. The open part of the lexicon gained 59 entries during the development rounds. Combined with the closed part, the final lexicon contained 165 entries. This lexicon was later replaced by a more extensive one, containing over 8000 entries (see Section 3.5).

5 Conclusions and Future Work

We have developed SimpleNLG-NL, a new version of SimpleNLG that is fit for surface realisation in Dutch. During the development process, the coverage of SimpleNLG-NL was gradually expanded by iteratively generating and testing on sentences from a Dutch treebank. Eventually, over 80% of the test sentences could be generated correctly, with a few acceptable differences in punctuation and word order. The issues with word order of auxiliary verbs will be addressed in future work.

Currently, word order can be altered with the use of premodifiers and postmodifiers. However, a better approach may be the one used in SimpleNLG for German, where Bollmann (2011) provided a feature to choose the desired word order. This also allows for easier sentence manipulation.

As the target sentences used for development and testing covered many different sentence structures, we believe the current grammatical coverage of SimpleNLG-NL is sufficient for simple surface realisation in Dutch. SimpleNLG-NL will be used in the POSTHCARD project[9] to realise (parts of) templates for dialogue generation, used to simulate conversations with patients suffering from Alzheimer's disease. The simulation aims to provide training for caregivers based on scenarios with a virtual Alzheimer's patient.

SimpleNLG is publicly available on Github.[10] Like SimpleNLG, SimpleNLG-NL is released under Mozilla Public License 1.1,[11] allowing for modification and commercial use. The SimpleNLG-NL code includes comments and Javadoc information that should make it easy to use and adapt. In addition, the SimpleNLG wiki[12] will be adapted for SimpleNLG-NL.

Acknowledgements

This research was carried out partially within the POSTHCARD project, funded by the European AAL programme (aal-call-2017-045). POSTHCARD is being made possible in The Netherlands by ZonMw, under project number 735170004.

References

Marcel Bollmann. 2011. Adapting SimpleNLG to German. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG 2011)*, pages 133–138.

Geert Booij and Jenny Audring. 2018. Separable complex verbs (SCVs). Retrieved June 04, 2018 from http://www.taalportaal.org/taalportaal/topic/pid/topic-13998813296768009.

Gosse Bouma, Gertjan Van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. In *Computational Linguistics in the Netherlands 2000: Selected Papers from the Eleventh CLIN Meeting*, pages 45–59.

[9] http://posthcard.eu

[10] https://github.com/rfdj/SimpleNLG-NL

[11] https://www.mozilla.org/en-US/MPL/

[12] https://github.com/simplenlg/simplenlg/wiki

Rodrigo De Oliveira and Somayajulu Sripada. 2014. Adapting SimpleNLG for Brazilian Portuguese realisation. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 93–94.

Albert Gatt and Ehud Reiter. 2009. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93.

Alessandro Mazzei, Cristina Battaglino, and Cristina Bosco. 2016. SimpleNLG-IT: Adapting SimpleNLG to Italian. In *Proceedings of the 9th International Natural Language Generation conference*, pages 184–192.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Pierre-Luc Vaudry and Guy Lapalme. 2013. Adapting SimpleNLG for bilingual English-French realisation. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 183–187.