Development of an Open Source Natural Language Generation Tool for Finnish

Mika Hämäläinen
University of Helsinki
Department of Modern Languages
mika.hamalainen@helsinki.fi

Jack Rueter
University of Helsinki
Department of Modern Languages
jack.rueter@helsinki.fi

Abstract

We present an open source Python library to automatically produce syntactically correct Finnish sentences when only lemmas and their relations are provided. The tool resolves automatically morphosyntax in the sentence such as agreement and government rules and uses Omorfi to produce the correct morphological forms. In this paper, we discuss how case government can be learned automatically from a corpus and incorporated as a part of the natural language generation tool. We also present how agreement rules are modelled in the system and discuss the use cases of the tool such as its initial use as part of a computational creativity system, called Poem Machine.

Tiivistelmä

Tässä artikkelissa esittelemme avoimen lähdekoodin Python-kirjaston kieliopillisten lauseiden automaattista tuottamista varten suomen kielelle. Kieliopilliset rakenteet pystytään tuottamaan pelkkien lemmojen ja niiden välisten suhteiden avulla. Työkalu ratkoo vaadittavan morfosyntaktiset vaatimukset kuten kongruenssin ja rektion automaattisesti ja tuottaa morfologisesti oikean muodon Omorfin avulla. Esittelemme tavan, jolla verbien rektiot voidaan poimia automaattisesti korpuksesta ja yhdistää osaksi NLG-järjestelmää. Esittelemme, miten kongruenssi on mallinnettu osana järjestelmää ja kuvaamme työkalun alkuperäisen käyttötarkoituksen osana laskennallisesti luovaa Runokone-järjestelmää.

1 Introduction

Natural language generation is a task that requires knowledge about the syntax and morphology of the language to be generated. Such knowledge can partially be coded

The source code is released in GitHub https://github.com/mikahama/syntaxmaker

Proceedings of the 4th International Workshop for Computational Linguistics for Uralic Languages (IWCLUL 2018), pages 51–58,
Helsinki, Finland, January 8–9, 2018.

by hand into a computational system, but part of the knowledge is better obtained automatically such as case government for verbs.

Having a computer create poetry automatically is a challenging task. Even more so in the context of a morphologically rich language such as Finnish which makes generating grammatical sentences, even when they are not creative, a challenge. Therefore having a syntactically solid system as a part of the poem generation process is extremely important.

In this paper, we present an open-source tool for producing syntactically correct Finnish sentences. This tool is used as a part of an NLG pipeline in producing Finnish poetry automatically. The poem generation part of the pipeline is out of the scope of this paper.

2 Related work

Previously in the context of poetry generation in Finnish (Toivanen et al., 2012), the problem of syntax has been solved by taking a ready-made poem, analyzing it morphologically and replacing some of the words in it, inflecting them with the morphology of the original words. This, however, does not make it possible to generate entirely new sentences, and it fails to take agreement or government rules into account, instead it expects agreement and government to be followed automatically if words with sufficient similarity are used in substitutes.

Another take on generating Finnish poetry in a human-computer co-creativity setting (Kantosalo et al., 2015) was to use sentences extracted from the Project Gutenberg's children's literature in Finnish. These sentences were treated as "poetry fragments" and they were used to generate poems by combining them together in a randomized fashion. This method indeed gives syntactically better results than the one described in (Toivanen et al., 2012), as it puts human-written sentences together, but it doesn't allow any variation in the poem apart from the order of the sentences in the poem.

Reiter (1994) identifies four different steps in an NLG pipeline. Those are content determination, sentence planning, surface generation, and morphology and formatting. In the content determination step, an input is given to the NLG system, e.g. in the form of a query to obtain desired information from the system. Based on this query, a semantic representation is produced addressing the results to the query. In other words, this step decides what information is to be conveyed to the user in the final output sentence, but also how it will be communicated in the rhetorical planning of the sentence.

The sentence planner will then get the semantic representation as input and produce an abstract linguistic form which contains the words to be used in the output and their syntactic relations. This step bears no knowledge of how the syntax will actually be realized, i.e., agreement or government rules, instead it applies the chosen words and how they are related to one another.

The last two steps of the pipeline deal with the actual realization of the syntax in the sentence. It is the task of the surface generator to handle the linguistic expression of the abstract linguistic structure. It means resolving agreement, forming questions in a syntactically correct manner, negation and so on. The actual word forms required are produced in the morphology step.

3 The Finnish NLG tool

The tool, Syntax Maker, described in this paper focuses on the surface generation step of the NLG pipeline described by Reiter (1994). It is used as a part of a complete NLG pipeline for producing Finnish poetry and is currently in place in the Poem Machine[1] system. This tool was made as a part of the poem generation system in order to solve the problem of creating novel, grammatical sentences not tackled by the previous Finnish poem generators. Taking an NLG point of view hasn't been studied before in the case of Finnish poetry, which is a shame since Finnish, unlike English, has a rich morphosyntax. This rich morphosyntax must be given proper attention if the computational creativity system is to be given more freedom to produce sentences of its own, using its own choice of words in a sentence that might cause other words around them to undergo morphological change as dictated by agreement and government.

Syntax Maker only knows the morphology needed in the level of tags. For example, it knows what case to use for a noun and what person to use for a verb. Actual morphological forms are generated using Omorfi (Pirinen et al., 2017).

3.1 Syntactic representation

Syntax Maker is designed to take the abstract linguistic structure of a sentence as its input. This structure consists of part-of-speech specific phrases each of which have their head word in lemmatized form. The phrases are nested under each other so that the highest possible root of the tree is a verb phrase.

When the phrases are nested together, they need to be added in proper slots to fulfill the requirements of agreement and government. For example, a noun phrase that is to act as a direct object of a verb phrase has to be nested in the verb phrase slot *dir_object*. In dealing with verb phrases, Syntax Maker automatically deduces the possible slots based on the verbs used as heads. In other words, Syntax Maker, determines the valency of a verb automatically and assigns values such as transitive, ditransitive or intransitive. On an abstract level, the phrases and their structures are defined manually.[2]

Using phrase structures gives us an easier way to implement the needed functionalities. Since the structure of phrases is similar for different parts-of-speech, we can reuse the same code across different parts-of-speech. The division into part-of-speech specific phrases gives us more freedom in expressing their peculiarities such as agreement and government rules and what kind of phrases can be nested under them. These structures come with a predefined word order, but it's not enforced by Syntax Maker. In other words, the word order in a phrase can be shuffled at will without losing the government or agreement information. Even with an altered word order, Syntax Maker can resolve the proper morphology correctly. The phrase structures are defined in JSON outside of the source code of Syntax Maker written in Python.

3.2 Handling government

Case government rules for adpositions have been hand coded. This can be attributed to the fact that there is only a very limited number of adpositions in Finnish, and it takes little time for a native speaker to write down the case required of a noun

[1] `http://runokone.cs.helsinki.fi/`

[2] These structures are available on `https://github.com/mikahama/syntaxmaker/blob/master/grammar.json`

phrase when it serves as complement to a given adpositional phrase. The analogous treatment of verbs, however, would be overly time consuming and laborious, and hence this has been automated.

As Finnish is an accusative language, the object is marked with a specific case. The case used depends on the verb in question and thus has to be specified for each verb separately. We obtain the case government information together with verb transitivity automatically from The Finnish Internet Parsebank (Kanerva et al., 2014) syntactic bi-grams.

Each line of the automatically parsed Parsebank bi-gram data consists of two word forms connected by a syntactic relation in the order in which they appeared in the sentence. These word forms are accompanied by their lemma, part-of-speech, morphology and syntactic annotation.

To extract the cases in which nouns have been linked to verbs, we look for lines in which the first word form has *V* as its part-of-speech tag and the second word form has *N* part-of-speech and *NUM_Sg* in its morphology. The reason why we limit the search to singular nouns only is that, in Finnish, a verb that takes its object in genitive in singular, takes it in nominative in plural, e.g. *syön kakun* and *syön kakut* but not **syön kakkujen*. Therefore taking plural objects into account as well would introduce more undesired complexity. Furthermore, we ignore all nouns where the lemma and word from are the same. This is done because the noun in question would then either be in the nominative, which is not an object case, or it will have been given an improper analysis in which case no lemmatization has been performed. Examples of this kind of wrong analyses in the corpus are *kattella/kattella/N/NUM_Sg* and *kasteleen/kasteleen/N/NUM_Sg*. For each bi-gram filling these criteria, we store the lemma of the verb and the case the related noun was in. This gives us a dictionary[3] of verbs and frequencies for noun cases associated with each verb.

The resulting dictionary is then used to determine the transitivity of a verb and the most frequent case for its object(s). This dictionary consists also of a plethora of non-verbs such as *Ljubuški* and *Dodonpa* as a result of erroneous parsing in the Parsebank data. This, however, causes no problems in the system because the dictionary also contains a extensive number of real, lemmatized verbs. Given that Syntax Maker operates on the level of surface generation, it is not actively involved in choosing the words in the NLG task. This means that, unless Syntax Maker is specifically instructed to use a non-verb it happens to know as a verb, it won't. This noise in the verb noun case dictionary, however, has no real effect on the grammaticality of the generated sentences.

The transitivity and most frequent case of the object is determined for a given verb by the verb noun case dictionary. The system is coded to accept the genitive, partitive, elative and illative as possible direct object cases and the essive, translative, ablative, allative and illative cases as indirect object cases. When the system defines whether a verb can take a direct object, it requires the relative frequency of one of the direct object cases to be above 23% of all the possible cases the verb has been seen with. For ditransitivity, the threshold is 18% for an indirect object case. Ditransitivity will not be considered if the verb is determined not to have a direct object. These threshold values have been adjusted by hand after looking at the performance of the system with a handful of verbs used in testing.

The genitive serves another use in Finnish syntax in addition to marking the direct

[3] The verb-noun case dictionrary is released on `https://github.com/mikahama/syntaxmaker/blob/master/verb_valences_new.json`

object. If the most frequent direct object case is genitive, we preform an additional check to see that it really is being used as an object function. The verb has to also has enough partitive case, over 23%, so that we can safely say that genitive indeed can be used as an object. This is because in Finnish, verbs that take their direct object in the genitive, also accept partitive in certain contexts such as in the expression of differences in aspect or negation.

3.3 Modelling agreement

Agreement, unlike government, is something that does not need to be extracted from a corpus. It is a rather straightforward thing and can be modelled with hand-written rules. In Finnish the predicate verb agrees in person and number with the subject, and adjective attributes agree in case and number with the head noun.

Since all the phrase types in our system are modelled in a similar way, it is easy to introduce agreement rules in the phrase structures. In a phrase structure, we define a key that is either *parent* referring to the parent phrase of the current phrase or a key to the list of *component*. Component lists all the possible syntactic positions for nested phrases such as *subject* or *dir_object*. Even though there aren't many agreement relations in Finnish, by modelling them in the external grammar file, we hope to make it easier to add more languages to the system in the future.

When Syntax Maker produces a sentence, it starts to process the syntactic tree phrase by phrase. For each phrase, it looks at the defined agreement relationship and copies the morphological information from the phrase defined to be agreed with. The agreement relation in the grammar file states the morphological tags which should be copied, for example in the case of an adjective phrase, the tags are *CASE* and *NUM*.

3.4 Modifying the verb phrase

Apart from just providing basic grammaticality by resolving agreement and government, Syntax Maker also provides means to modify verb phrases to produce more complex, yet grammatical sentences.

Syntax Maker can be used to negate sentences. When a sentence is negated, a new phrase with the head *ei* is added to the components of the verb phrase as *aux*. The new phrase has an agreement relation *parent->subject: PERS, NUM* and the verb phrase containing the predicate verb is tagged as *NEG* so that it will be conjugated as such when the full sentence is produced as text. The case of the direct object is also changed to partitive if the most frequent direct object case of the verb is genitive, in compliance with Finnish grammar.

Mood and tense are also handled by Syntax Maker. In the case of the prefect, the auxiliary verb *olla* is set as the new head of the verb phrase and the old head is moved to a new subordinate phrase with the part-of-speech value *PastParticiple* and agreement *parent->subject: NUM*. This makes sense from the point of view of Syntax Maker since *olla* is the verb that is conjugated normally while the participle form only agrees with the number of the subject. Other auxiliary verbs can be added in a similar fashion, where the auxiliary verb substitutes the original head and the verb is moved to a nested phrase with the morphology required by the auxiliary verb.

Passive voice is handled by creating a dummy phrase as a subject with the morphological tags *PERS = 4* and *NUM = PE*. This will automatically make the verb agree with the dummy phrase's morphology and produce the correct form as output. Also,

if the verb takes its direct object in genitive, the government rule is changed so that the direct object will be in the nominative.

A sentence can also be turned into an interrogative one. This adds an additional morphological tag *CLIT = KO* to the head of the verb phrase and moves it to the beginning of the whole sentence. Syntax Maker does not produce punctuation, so a question mark has to be appended to the end of the sentence at a different level in the NLG pipeline.

4 Evaluation

In this part, we evaluate how accurately Syntax Maker can produce verb phrases. We limit this evaluation to the automatically extracted information used by Syntax Maker because it is more prone to errors than the hand written rules. This means that we are evaluating two things in the generated output: the predicted valency i.e. how many objects the verb can take and the predicted case for the object.

In order to do the evaluation, we take a hundred Finnish verbs at random from the Finnish Wiktionary[4]. These verbs are then given as input to Syntax Maker to produce verb phrases out of them. The valency and object cases are then checked by hand to conduct the evaluation phase.

	too low	**too high**	**correct**
valency prediction	28%	5%	67%

Table 1: Accuracy in predicting valency

Syntax maker predicts the number of objects correctly 67% of the time and 28% of the time too low. This is acceptable in the task of poem generation where we are interested in generating syntactically correct poems. Having too few objects in the generated output only creates an ellipsis that doesn't result in incorrect syntax. However, in other NLG tasks outside of the scope of poem generation, the objects might be important and thus having a higher accuracy is something to work towards.

	case correct	**case incorrect**	**no object**
object case prediction	50%	4%	46%

Table 2: Accuracy in predicting object case

In the test set, Syntax Maker produced a wrong case only 4% of the time. 46% of the verbs were either truly intransitive or didn't take an object according to Syntax Maker. In other words, by just taking into account the transitive verbs recognized by Syntax Maker, the accuracy reached to 93%. This means that Syntax Maker is very good at coming up with the correct case but not as good at determining the valency accurately.

5 Future work

At the current state, Syntax Maker doesn't handle all parts of the Finnish grammar. For instance, it doesn't have the functionality to express aspectual difference by alter-

[4] From a Wiktionary dump on `https://dumps.wikimedia.org/fiwiktionary/`

ing between genitive and partitive objects. In addition, it has only a limited knowledge of the transitivity of verbs. Novel automated ways should be studied to solve this shortcoming.

In the future, Syntax Maker should be tested as a part of the NLG pipeline in uses other than poetry generation as well. This might reveal new requirements for the system that do not appear in the task of poetry generation. This might also reveal missing functionalities both in the generation of syntax and the API provided by the library that are needed in other NLG tasks.

Including small Uralic languages in this tool is also in our interest for the future. This is because having an NLG system would be especially useful in the case of minority languages, for example in generation of news automatically in these languages.

6 Conclusions

In this paper we have presented an open source Python library called Syntax Maker. The library was made to be used as a low-level syntax producer in a new NLG pipeline for producing Finnish poetry and is currently in place in a computational creativity system known as Poem Machine[5]. By embracing the notion of separation of concerns in the software architecture of the system, Syntax Maker can be used in a multitude of contexts outside of computational creativity applications as an all-purpose tool for producing grammatical Finnish. To achieve this goal, a method for extracting the information needed to resolve verbal agreement automatically was presented and evaluated.

Acknowledgments

This work has been supported by the Academy of Finland under grant 276897 (CLiC)

References

Jenna Kanerva, Juhani Luotolahti, Veronika Laippala, and Filip Ginter. 2014. Syntactic n-gram collection from a large-scale corpus of internet finnish. In *Proceedings of the Sixth International Conference Baltic HLT*.

Anna Kantosalo, Jukka Toivanen, and Hannu Toivonen. 2015. Interaction evaluation for human-computer co-creativity: A case study. In *Proceedings of the Sixth International Conference on Computational Creativity*. pages 276–283.

Tommi A Pirinen, Inari Listenmaa, Ryan Johnson, Francis M. Tyers, and Juha Kuokkala. 2017. Open morphology of finnish. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11372/LRT-1992.

Ehud Reiter. 1994. Has a consensus nl generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation*. INLG '94.

[5] Poem Machine can be used on `http://runokone.cs.helsinki.fi/`

Jukka Toivanen, Hannu Toivonen, Alessandro Valitutti, and Oskar Gross. 2012. Corpus-based generation of content and form in poetry. In *Proceedings of the Third International Conference on Computational Creativity*.