Harnessing NLG to Create Finnish Poetry Automatically

Mika Hämäläinen
Department of Digital Humanities
University of Helsinki
Fabianinkatu 33, 00014 Helsinki Finland
mika.hamalainen@helsinki.fi

Abstract

This paper presents a new, NLG based approach to poetry generation in Finnish for use as a part of a bigger Poem Machine system the objective of which is to provide a platform for human computer co-creativity. The approach divides generation into a linguistically solid system for producing grammatical Finnish and higher level systems for producing a poem structure and choosing the lexical items used in the poems. An automatically extracted open-access semantic repository tailored for poem generation is developed for the system. Finally, the resulting poems are evaluated and compared with the state of the art in Finnish poem generation.

Introduction

Creating poems automatically is a difficult task to tackle, especially since poetry as a genre is fragmented and not easily defined (Juntunen 2012). What makes computer generated poetry more difficult than a traditional NLG task is that poems usually express their meaning in an indirect language by means of different rhetorical devices, such as, metaphors, ellipsis, rhymes and so on. These are issues to resolve above and beyond the mere production of grammatical output that follows the syntax of the target language.

Grammaticality is a big issue especially in the case of morpho-syntactically rich languages such as Finnish. Finnish has a set of agreement and government rules in its syntax, which means that words in a sentence affect each other's morphological form depending on their syntactic relations. In other words, where as in English it is almost possible to produce grammatical output just by using words in their dictionary forms (i.e. lemmas) in a sentence with slight to no modifications at all, in Finnish, more often than not, words have to be inflected to fit the morphological requirements of the syntax in the sentence.

One of the tasks we have to solve in order to produce grammatical poems with an NLG pipeline is to build a linguistically robust system for resolving the morpho-syntax of the words in a sentence. Due to the lack of freely available systems of this kind for Finnish, we have to build one of our own. On top of this syntax producing system, we then build higher level functionality to produce poetry.

In addition to solving the challenging problem of grammaticality, in this paper, we also present a way of building a semantically linked database to use in computationally creative systems. This database consists of syntactic relations between words, which also reveals a great deal about their semantics and the intercompatibility of their meaning in a sentence. Finally, on top of the syntax generator and this semantic database, we propose a method for generating modern poetry in Finnish.

The key notion behind the system is that if the output is grammatical, i.e. the language is good, the structure resembles that of a poem and there is semantic cohesion within the poem, the poems produced by the system will be accepted as poetry by people. In this paper, we discuss how we solved these different requirements for the poetry produced.

The poem generator described in this paper replaces the previous poem generator in a computationally creative system called the "Poem Machine[1]". The generator then serves as a component in a larger whole of a system with a focus on providing an environment for human-computer co-creativity. The Poem Machine is intended to be used in elementary schools to aid school kids in writing poems of their own by removing the problem of "a blank page" and by offering a computer-generated poem as a starting point for poem writing. This paper, however, focuses solely on the poem generator as an independent component of the system leaving the co-creativity aspect of the larger system outside of the scope of this paper.

Related work

Poem generation has received its fair share of interest in past research. The problem has been approached from different angles such as translation with weighted finite-state transducers (Greene, Bodrumlu, and Knight 2010), text transformation via word embeddings (Bay, Bodily, and Ventura 2017), templates (Colton, Goodwin, and Veale 2012) and case-based reasoning (Gervás 2001) among others. In this section, we describe the approaches used in the context of Finnish poetry generation in more detail.

The current state of the art in Finnish poetry generation is the P. O. Eticus system (Toivanen et al. 2012). Unlike in our approach, P. O. Eticus does not employ a linguistically solid NLG surface generator to produce syntactic Finnish, instead it takes its syntax from existing poems. The way the system

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

works is that it takes a ready-made human written poem at random, analyzes it morphologically and swaps some words in the poem with new ones making sure that the morphology, such as case and number, matches that of the words in the original poem that are to be replaced.

From a linguistic perspective, this means that the grammaticality of the final poem relies on pure chance. By making sure that the morphology matches, the system is able to solve agreement of nouns and adjectives, but will fail if the government rules for the new words are different from the ones that existed in the poem. This also means that the subject of the sentence cannot be changed at will into a different person or number without producing ungrammatical output, let alone the fact that the system is incapable of producing any syntactic structure of its own, as it relies heavily on the structure of the ready-made poems.

Another take on the poem generation in Finnish is that of (Kantosalo, Toivanen, and Toivonen 2015). This is by no means a sophisticated approach as it takes poem fragments from thematically divided children's literature at random and puts them together without any further analysis of their semantics or meter. However, it is important to showcase this approach as it was the original one in use in the Poem Machine, which nowadays uses a generator based on the one we are describing in this paper. The poems produced in this way, were more grammatical than the ones of P. O. Eticus because the verses were never altered, but they had less semantic cohesion because the verses came from very different sources, although all of them shared a keyword.

The NLG Pipeline

In this paper, we propose an NLG pipeline with independent modules to tackle different problems of generating Finnish poetry. Syntax, morphology, semantics and poetic structure are all different parts of the system. This makes it possible to separate the two goals for the generated poems at the level of implementation, grammaticality and semantic cohesion.

An NLG pipeline has traditionally been divided into four different steps: content determination, sentence planning, surface generation, and morphology and formatting (Reiter 1994). This is also the definition for NLG we are following in this paper.

In the content determination step, an input such as a query is fed to the system based on which the output will be produced. This might be for example weather for a particular city requested by the user. The results of the query are in a form of a semantic representation of the information that will be conveyed to the user in the final output. In addition to what the information will be in the final output, this step also tackles the question of how it should be communicated such as in the form of rhetorical planning.

In sentence planning, the abstract semantic representation is further processed into an abstract syntactic representation. This means that the lexical items and their syntactic relations are chosen by the system. This step works on a high level of abstraction of syntax and it does not require knowledge of how the syntax is actually produced. This means that it has no knowledge of government or agreement rules in the language, which, as we will see later in this paper, are extremely important features in the Finnish grammar.

The surface generator is the one in charge of resolving the actual syntax needed to express the abstract syntax relations (such as subject, predicate, object) by following the grammar rules of the language in question. For example, in Finnish the predicate has to agree with the number and person of the subject, and the object has to be in a case governed by the predicate verb.

Finally, the syntactic knowledge is passed to a morphological generator which is the one in charge of inflecting the lexemes chosen in the previous steps based on the lemma and the morphological features resolved by the surface generation step.

In this paper, we will focus on the two higher level steps of the NLG pipeline in *the Poem Generator* section. For the surface generation part, we have developed a tool called Syntax Maker[2] (Hämäläinen 2018) due to the absence of openly available NLG tools for Finnish. Syntax Maker is only briefly discussed in this paper, for a more detailed description and evaluation has already been published elsewhere (Hämäläinen and Rueter 2018).

The reason why Finnish requires a sophisticated tool for generating the syntax of the sentence lies in the highly agglutinating nature of the language. In Finnish, syntactic roles of words are not expressed by strict word order as is the case in English, but rather by inflecting words accordingly to their syntactic function. There are two syntactic rules that affect the word forms in Finnish: agreement and government.

Agreement in the Finnish context means that the predicate verb has to agree in number and person with the subject and that adjective attributes have to agree in case and number with the noun they modify. This has been solved by rules in the implementation of Syntax Maker.

Government is a more difficult phenomenon. Usually, in Finnish, verbs take their direct object either in the genitive (or the accusative for the personal pronouns) or the partitive. This cannot be deduced by easy rules, but rather has to be known for each verb individually. This was achieved by learning the cases of the objects (direct and indirect) automatically from the Finnish Internet Parsebank (Kanerva et al. 2014) and its syntactic bi-gram data.

The morphological generation is done by using Omorfi (Pirinen et al. 2017), which is a finite-state based tool for analyzing and generating the morphology of Finnish words. The morphological forms together with lemmas resolved by Syntax Maker are given to Omorfi, which inflects the words accordingly to its rules and the input.

The Semantic Repository

Whereas P. O. Eticus uses a graph of semantically similar words obtained by connecting words together with a log-likelihood ratio test (LLR), we want not only to capture the overall semantic relatedness of the words but also word relations in their syntactic position. In other words, we do not

[2]Syntax Maker is released as an open source Python library on https://github.com/mikahama/syntaxmaker

simply want to build our network so that we can deduce that *dog* and *dog food* are related based on their shared context, but rather that they are related by virtue that *dog* is capable of performing the action *eat* and *dog food* can serve as a direct object for such an action.

In order to capture both the semantics and syntax, we build our semantic repository[3] so that it contains lists of lemmatized words by their parts-of-speech. These lists are interconnected to a network based on the syntactic relations these words have had in a corpus. The strength of the connection is determined by the frequency of co-occurrence of the words in a given syntactic relation revealing more about the semantic relatedness of the words. To achieve this, a syntactically parsed corpus of Finnish is needed.

As a corpus for extracting the semantic knowledge, we use the Finnish Internet Parsebank (Kanerva et al. 2014), and more specifically its data-set of syntactic bi-grams. The corpus is one of the largest syntactically parsed ones in Finnish consisting of 116 million sentences and 1.5 billion individual tokens. The text of the corpus originates from different sources found on the internet by Common Crawler[4].

The data has been automatically parsed into syntactic dependency trees, and the syntactic bi-gram data consists of bi-grams of words that have appeared next to each other in the syntactic tree. This means that as opposed to a traditional bi-gram, it is perfectly possible that words that have not been immediate neighbors in the sentence, but are related to each other by one arc in the syntactic tree, appear in the bi-gram list. In other words, for example, a noun acting as a direct object of a verb will appear in the syntactic bi-grams even if in the actual sentence there was an adjective in between the verb and the noun.

We build our semantic repository based on the co-occurrences of the words in the syntactic bi-grams. Since the data-set has been parsed entirely automatically, however, the data is not free of noise. This is why, we define additional requirements for the two words of a bi-gram before we update the relation to our semantic repository.

For verbs, we use the syntactic knowledge in Syntax Maker to perform an additional check. For each verb found in the bi-grams, we query Syntax Maker for the valency, in other words, how many objects the verb can take, and the cases of the objects. We only update the verb-object relations to the repository if the noun has been inflected in a case that is possible for the verb in question, given its case government.

For subjects, it is not necessary to query the Syntax Maker because in Finnish, the subject is, almost without an exception, always in the nominative case, which means that we can check that directly. Additionally the verb has to be in active voice, because the direct genitive object of a verb in passive voice appears in the nominative, which, if not filtered out, would introduce more noise in the data.

A noun-adjective relation is only considered if the noun and the adjective share the same grammatical case. This is due to the Finnish agreement rule which requires the case of an adjective attribute to agree with the case of the noun.

For other syntactic relations such as adverb to verb relation, we don't specify any further constraints based on the compatibility of the words according to their morpho-syntax. This couldn't even be achieved, for example, in the case of adverbs and verbs, as there are no agreement or government rules for them in the Finnish grammar.

By imposing restrictions of morpho-syntax, we were able to solve a part of the issue caused by the noise in the data, that is, the true syntactic relatedness of the words without erroneous relations introduced by the parser. The syntactic parser, however, was not the only source of errors in the data. The data also contains a multitude of words that are incorrectly tagged, for example the adjective in its partitive form *esiintulevaa* (appearing) was incorrectly tagged as a verb. Also, the corpus contains non-words such as *her?nnyt* (awoken) with an encoding error, and non-lemmatized word forms as lemmas such as the optative *heitetääs* (let's throw) where the correct lemma would be the infinitive *heittää* (throw).

In order to remove incorrectly tagged words from our syntactic repository, we go through all the words with Omorfi. Omorfi produces all the possible morphological interpretations for its input word form. In the case of Finnish, this usually results in a long list of possibilities because the inflected forms of Finnish are frequently homonymous. For example, the word form *voi*, is interpreted by Omorfi as a possible form of *voi* (butter), *voida* (can), *voitaa* (to butter), *voittaa* (to win) or *vuo* (flow).

There are two things we look at in the list of possible lemmas produced by Omorfi for each word. First, we look to see if at least one of the possible interpretations has the same part-of-speech reading as recorded in our semantic repository and if the lemma of at least one of the interpretations with the same part-of-speech is the same as the word in the repository. If the part-of-speech does not match, or the word is not in a lemmatized form, we remove it from the repository. Because Omorfi is a rule based system and we are not using a guesser version of it, it makes no attempt to analyze anything it does not know. This also allows us to filter out the non-words resulted mostly from encoding errors or spelling errors.

After the extraction and filtering process, our semantic repository consists of over 9569 adverbs, 18300 verbs, 5900 adjectives and 965000 nouns that are connected to each other by the syntactic relation they have shared in the corpus weighted by the frequency they have appeared together in that syntactic relation. The high number of nouns is due to the Finnish orthographic rule of writing compound nouns together as one word, where as in English they would be written separately, for example the English *gas station* and its Finnish translation *huoltoasema*, which consists of two words *huolto* and *asema*.

The Poem Generator

In this section, we describe how the higher levels of the NLG pipeline, namely content determination and sentence planning, are implemented in the system. These operations rely

[3]The semantic repository can be browsed and downloaded on https://mikakalevi.com/semfi/

[4]http://commoncrawl.org/

mainly on the semantic repository as their main source of data, but the Finnish WordNet (Lindén and Carlson 2010) is also used as an additional data-set.

The actual poem generation part is divided into multiple diverse verse generators. Each one of these verse generators is only in charge of producing one verse of the poem. Semantic cohesion is achieved by the fact that each verse generator takes a noun as its input and outputs a noun together with the produced verse. This noun is then fed to the next verse generator. Some of the verse generators do not modify the noun while others change it. This way the verse generators produce a semantically coherent whole.

Each verse generator implements its own content determination and sentence planning steps. Regardless of the verse generator, the content determination starts with the input noun passed to it either by the user or the previous verse generator. This noun is used to look up related words in the semantic repository and additionally in the Finnish WordNet. Each verse generator has an abstract definition of the syntax of the verse in the form of syntactic relations that can be expressed by the verse. This is the sentence planning part of the generators. The actual realization of the grammar is done by feeding this information to Syntax Maker.

The decision which verse generators will be used and in which order is defined by the poem structure database. The database consists of a set of hand-coded poem structures. These structures only state the generators and their order, but they do not affect the functionality of each individual generator in any way. In theory, the verse generators and their order could also be randomized or decided automatically in a more justifiable way than pure random, but for now, we have opted for an approach involving predefined structures to ensure a higher structural coherence within the poem.

Verse Generators

In this section, we will explain the functionality of each individual verse generator in the system. There are altogether 12 different verse generators implemented in the system.

Metaphor Generator The content determination for the metaphor starts by looking at the list of verbs the input noun can act as a subject for. We want to construct a metaphor of a form *X is Y* where the two nouns, *X* and *Y*, are connected together by a verb that is semantically strongly related to both of them. The metaphor generator will also output a second verse right after the metaphorical one explaining the metaphor by revealing the verb.

By trying out different values, we reached to two threshold values for the frequency of the subject relation between the noun and the verb in the semantic repository. A verb is considered for metaphor production if it has occurred at least 20 times and at most 1000 times with the noun. This way, we can filter out verbs that aren't descriptive for the noun because they are too frequent in general and also verbs that don't co-occur often enough.

The next step is to list other nouns that can act as subjects for these verbs. These other nouns are also checked for the frequency of their subject relation to the verb. This yields us lists of possible nouns we can use in the metaphor together with the linking verbs. This, however is not enough, because now these lists will contain a lot of nouns that are semantically too similar. For example, *man is a woman* would be a frequently appearing metaphor because both of the nouns have a lot of verbs in common. This is why we remove all the nouns that have more than 5 verbs in common with the input noun we are searching metaphors for. This results in a list of nouns that are far enough semantically from each other.

Out of the obtained noun-verb list, we pick a noun and a verb at random to form the metaphor. For the second, metaphor explaining, verse we look at the number of objects the selected verb can have and fill the object slots with weighted random based on the object relations and their frequencies with the verb in the semantic repository. If the verb doesn't take an object, an adverb related to the verb is picked with the same weighted random approach as in the case of objects.

Rakkaus on luovuus	Love is creativity
Se kukoistaa ajan	It blossoms for a while
Viha on tapa	Hatred is a habit
Se ruokkii ajattelua	It provokes thought

The examples given above are possible outputs from the metaphor generator. The structure of the sentence planner is predefined as it is passed to the surface generator. The metaphor generator passes the newly picked noun to the following verse generator.

Synonym in Essive This verse generator creates sentences of a type *As a synonym, it does something*. How this is done, is that a synonym is looked up for the input noun in the Finnish WordNet by using NLTK (Bird, Klein, and Loper 2009). This is done by querying all the possible synsets for the noun and getting the lemmas for them. Then, a noun is picked at random from this synonym list to appear in the essive case in the sentence. The essive corresponds to *as a noun* in English.

The verb appearing in the verse is again looked up in the semantic repository. The verb is picked based on the noun obtained from the WordNet so that the noun can function as its subject at weighted random. The complements of the verb are filled based on the valency of the verb and the objects linked to it in the semantic repository. This produces verses of the following kind.

Passipoliisina se uskoo laatuun	As a patrol officer, he believes in quality
Konnana se rötöstelee	As a crook, he nicks

The first example was generated for the input noun *police* and the second one for *dog*. It's important to note that the synonyms coming from the Finnish WordNet might not always be truly synonyms such as *dog* and *crook* because the Finnish WordNet has been translated from the English one directly and sometimes the Finnish translations are quite far from the English originals. This generator passes the WordNet synonym to the next generator.

Rhetorical Question This generator looks up an adjective for the input noun in the semantic repository based on the noun-adjective attribute relationship. The adjective is picked at weighted random based on the frequency of co-occurrenceIn addition, an interrogative pronoun is picked at random to form the question. The idea behind this question forming generator is that if we know that the noun has the adjectival property we are forming the question of, we can presuppose this. Instead of stating something is something, when we know that it'the case, we can just ask *why* that is the case, *how* and so on.

Milloin liekki on keltainen?	When is a flame yellow?
Miten paha on uhka?	How bad is the threat?

The examples illustrate verses generated for *flame* and *threat* respectively. The input noun is passed as such to the following verse generator.

Personal Pronoun Verses This category consists of four different generators. What they have in common is that they use a 1st or 2nd person personal pronoun either in the plural or singular as their subjects. The simplest one of them just looks up an adjective based on the input noun and forms a question of a type: *am I adjective?*.

There are two verse generators that are used to express an attitude. The first one forms a main clause with a personal pronoun subject and a verb that can be used to express an attitude, such as *hope* or *doubt*. The main clause can be turned into negative with a 50 % chance or additionally into a question with a 50 % chance. After producing the main clause, a subordinate clause is added to the main clause. The subordinate clause takes the input noun as its subject and then proceeds into looking for a suitable verb and objects and an adverb for it in the same manner as described for the previous verse generators.

The other attitude expressing verse generator picks a verb based on the pronoun picked to generate the verse. This is also done by a weighted random based on the subject connection of the verbs and the pronoun. Then it uses the input noun as an object for this verb. In order to express an attitude, an auxiliary verb is selected at random to be used in the sentence.

The last personal pronoun verse generator formulates a conditional subordinate clause in which the input noun is the subject and the verb and their objects are picked as seen before. The main clause has a verb in the conditional mood with a personal pronoun as its subject, the verb picked is selected on the basis of the input noun. This is done in such a way that the verb takes a noun that can act as its object according to the semantic repository.

Olenko huima?	Am I wild?
Emmekö me ajattele, ennen kuin nokkeluus pelastaa maan?	Won't we think before cleverness saves the earth?
Minä haluan noudattaa silkkiä	I want to follow silk
Vaikka savut houkuttaisivat onnenonkijaa, käyttäisitkö sinä savua?	Even if the smokes lured a fortune hunter, would you use smoke?

The examples above are output from the verse generators in the order of their presentation in this section. These are produced for the nouns *week*, *cleverness*, *silk* and *smoke* respectively. None of these verse generators alter the input noun, but rather pass it as it is to the following generator.

Paraphrase The paraphrasing verse generator expresses the input noun in other words by looking up a suitable hyponym or hypernym for it in the Finnish WordNet. In addition, an adjective is picked for the input noun from the semantic repository at weighted random. This adjective is used to describe the noun obtained from the Finnish WordNet in the verse.

Vesi, tuo ihmeellinen neste	Water, that wondrous liquid
Rukous, tuo hiljainen siunaus	Prayer, that silent blessing

The examples above are generated for *water* and *prayer*. The input noun is added to the beginning of the verse, separated by a comma. The verse generator passes the noun looked up in the WordNet to the next verse generator.

Relative clause This verse generator creates a relative clause which takes the subject position of the main clause in the verse. The object of the relative clause is a synonym for the input noun based on the Finnish WordNet. The object of the main clause is the input noun. The verbs for both clauses are looked up from the semantic repository based on the nouns they will have as objects.

Se, joka loppuu hauskuuteen, kertoo ilosta	What ends in fun, tells of joy
Se, joka lukee surun- valittelua, kuuluu suruun	Who reads condolence, forms part of the sorrow

The examples above are generated by using *joy* and *sorrow* as their input. The verse generator passes the input noun to the following generator unmodified.

Simple generators There are two extremely simple verse generators in the system. One is used to address a noun. What it does is that it outputs the input noun followed by a comma. The other simple verse generator generates tautologies, either in indicative or in potential, of a type *Xs are Xs*.

Poika,	Boy,
Pojat ovat poikia	Boys will be boys

The above examples are output for *boy* in both generators. Neither of them swaps the input noun, but rather passes it on as it is.

Example Poem

Here is an example poem to illustrate how the different verse generators can play together in a poem structure.

Usko on ihminen	Faith is a human
Se pelastuu kuolemasta	It is rescued from death
Henkilönä se upottaa veteen	As a person, it sinks in water
Miten arvoinen on henkilö?	How valuable is a person?
Minä en kartuta henkilöä	I don't accumulate a person
Olenko hieno?	Am I elegant?

Evaluation

In this section, we conduct an evaluation of the poems produced by the system. Evaluating poetry, even in the case of a human produced one, isn't an easy task, and it is something that is very difficult to do objectively. This is why we didn't want to come up with an evaluation metric of our own, rather we chose to use the same subjective evaluation metric that was used for P. O. Eticus (Toivanen et al. 2012). An additional advantage of this is that we can see how well our system fares in the same evaluation as the state of the art.

The P. O. Eticus was evaluated by 20 randomly picked university students. In order to have a better comparability of the results, we also randomly recruited university students for our evaluation. In the evaluation, we randomly selected 5 poems produced by our generator and 5 poems by the poem fragment approach (Kantosalo, Toivanen, and Toivonen 2015) which was previously in use in the Poem Machine. We had altogether 25 evaluators to go through poems produced by both systems. The order in which the poems were presented to the evaluators was randomized.

The evaluators were asked to evaluate texts rather than telling them directly that the texts are supposed to be poems. They weren't told that the texts they were reading were computer produced.

The evaluators were asked to answer to a binary question with a yes/no answer whether the text they were reading was a poem. In addition to that they were presented with six ad ditional questions: (1) How typical is the text as a poem? (2) How understandable is it? (3) How good is the language? (4) Does the text evoke mental images? (5) Does the text evoke emotions? (6) How much does the subject like the text? These questions were evaluated in the Likert scale from one (very poor) to five (very good).

As the evaluation questions are highly subjective and the evaluators' opinions on the poems vary a great deal, the results obtained for our approach and the poem fragment approach aren't directly comparable with those obtained previously for P. O. Eticus. However, the results P. O. Eticus got when it was evaluated are shown in the chart for reference purposes.

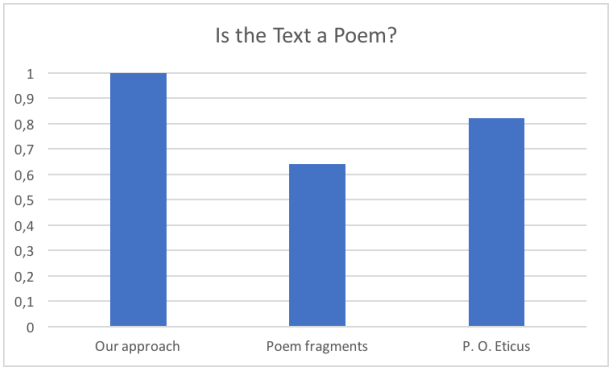

Figure 1: Results for the binary question

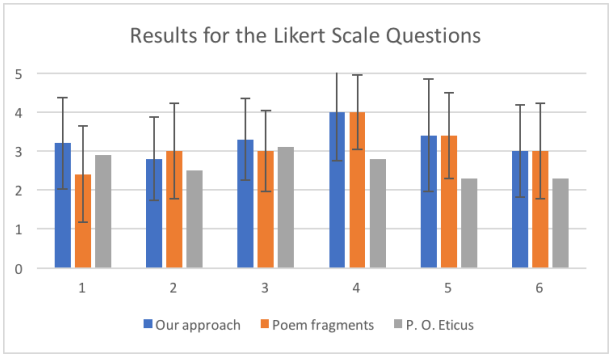

Figure 2: Results for the Likert scale questions with standard deviation: (1) How typical is the text as a poem? (2) How understandable is it? (3) How good is the language? (4) Does the text evoke mental images? (5) Does the text evoke emotions? (6) How much does the subject like the text?

Figure 1 represents the results obtained in the the first binary question whether the output of the generators was considered as a poem. An important finding is that the poem fragment approach only generates output recognizable as poetry 64 % of the time whereas the judges agreed unanimously that the output produced by our method is poetry. This is probably due to the fact that the poem fragment approach doesn't aim towards a poem-like structure whereas our approach uses predefined poem structures.

The results for the Likert scaled questions are shown in Figure 2. The results show that our approach outperformed the existing poem fragment generator in the typicality aspect and in how good the language was. Based on these results, we can deduce that our system is capable of producing poetry that is also accepted as poetry. Also the grammatical correctness of the output is high enough to score well against the fragment approach which essentially uses human written fragments.

Understandability is the only parameter in which the poem fragment approach performed better. This is could be due to the fact that the fragments are written by humans, which might contribute to them being easier to understand, where as the words picked by our system to be used in a

verse, might result in a sentence that is semantically more difficult to grasp.

The fact that our system seems to fare well in comparison with the state of the art on all aspects seems promising. However, since the poems generated by both systems were evaluated by different people, further study is needed to draw any final conclusions on which one actually performs better in this evaluation setting.

Discussion and Future Work

The generator discussed in this paper is a first step towards an NLG pipeline in poem generation in Finnish. Now that the most difficult parts of producing Finnish have been solved, namely the rich morphosyntax of the language, and that we are capable of producing grammatical Finnish from abstract syntactic representation, the next step is to reduce the hand-crafted nature of the verse generators. We are currently looking into the possibility of learning verse structures from real poetry into an abstract syntactic representation that we could fill, for instance, with the content determinators already defined for the individual verse generators. This would mean that we would only need to replace the sentence planning part of our pipeline to introduce more freedom into the system.

We could also extend the semantic repository not only to contain a wider list of syntactic relations but also to contain semantic data of a different nature. This could, for example, mean linking related words based on word embeddings. The extension of the semantic repository is a requirement we have already found in our initial experiments of using learned verse structures, because the syntactic relations in real verses are more complex than the ones modelled in our semantic repository.

Another important aspect for future research is studying this method in the context of the system for which it was initially built, namely the Poem Machine. Studies are currently underway on the co-creativity aspect of the Poem Machine in which the method described in this paper is in a collaborative setting with school kids assisting them in writing poetry of their own.

Conclusion

In this paper we have presented and evaluated an NLG approach for poem generation for the morphologically rich Finnish. The proposed approach is currently in use in a computationally creative system called Poem Machine which makes human computer co-creativity possible. The results of the evaluation seemed promising and we identified future directions for this research.

As a result of the study, an open source surface generation NLG tool for Finnish (Syntax Maker) was publicly released. Also, the syntactic repository data set has been made openly available for anyone interested in building their work on top of it.

Acknowledgments

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

I thank my PhD supervisor Jack Rueter for a keen eye on English grammar while reading this paper and his enthusiasm in bringing this approach to the minority language context.

References

Bay, B.; Bodily, P.; and Ventura, D. 2017. Text transformation via constraints and word embedding. In *Proceedings of the Eighth International Conference on Computational Creativity*, 49–56.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Colton, S.; Goodwin, J.; and Veale, T. 2012. Full-face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.

Gervás, P. 2001. An expert system for the composition of formal spanish poetry. *Knowledge-Based Systems* 14(3):181–188.

Greene, E.; Bodrumlu, T.; and Knight, K. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, 524–533. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hämäläinen, M., and Rueter, J. 2018. Development of an Open Source Natural Language Generation Tool for Finnish. In *Proceedings of the Fourth International Workshop on Computatinal Linguistics of Uralic Languages*, 51–58.

Hämäläinen, M. 2018. Syntax Maker. https://doi.org/10.5281/zenodo.1143056.

Juntunen, T. 2012. Kirjallisuudentutkimus. In *Genreanalyysi: tekstilajitutkimuksen käsikirja*, 528–536.

Kanerva, J.; Luotolahti, J.; Laippala, V.; and Ginter, F. 2014. Syntactic N-gram Collection from a Large-Scale Corpus of Internet Finnish. In *Proceedings of the Sixth International Conference Baltic HLT*.

Kantosalo, A.; Toivanen, J.; and Toivonen, H. 2015. Interaction Evaluation for Human-Computer Co-creativity: A Case Study. In *Proceedings of the Sixth International Conference on Computational Creativity*, 276–283.

Lindén, K., and Carlson, L. 2010. FinnWordNet-WordNet på finska via översättning. *LexicoNordica Nordic Journal of Lexicography* 17:119–140.

Pirinen, T. A.; Listenmaa, I.; Johnson, R.; Tyers, F. M.; and Kuokkala, J. 2017. Open morphology of Finnish. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Reiter, E. 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.

Toivanen, J.; Toivonen, H.; Valitutti, A.; and Gross, O. 2012. Corpus-Based Generation of Content and Form in Poetry. In *Proceedings of the Third International Conference on Computational Creativity*.