University of Warsaw
Faculty of Philosophy

Ryszard Tuora
Record book number: 351046

Dependency Trees in Automatic Inflection of Complex Phrases in Polish

Bachelor's thesis
in the field of Cognitive Science

The thesis was written under the supervision of
dr hab. Justyna Grudzińska-Zawadowska
and dr Łukasz Kobyliński
Faculty of Philosophy
University of Warsaw

Warsaw, December, 2022

Summary

This thesis discusses a method for combining different linguistic representations in order to create an automatic inflection system for complex phrases in Polish. First the main motivation for such a system is elucidated: the use of automatic inflection for template-based natural language generation. Subsequently the resources used, are introduced: NKJP tagset for representing the morpology of Polish words, and PDB UD dependency grammar for representing the syntactic structure of phrases. In the third section, the implementation details are fleshed out. The final solution recurrently decomposes the task of inflecting phrases, with the use of dependency trees. The base case is the inflection of individual words, which is approached twofold: by using a dictionary, and a *seq2seq* neural network. The recurrent algorithm invokes these methods for each node in the tree, with the parameters of each inflection being calculated based on using dependency relations as a proxy representation of accommodation relations. Evaluation of these methods on some handcrafted test cases, and a bigger dataset is provided. Some examples of applying the tools in a template-based NLG environment are shown. A handful of extensions of the system is discussed: basic support for particular derivation mechanisms, phrase lemmatization, and extensions into other languages.

Keywords

NLP, inflection, dependency grammar, morphology, NLG, machine learning

Title of the thesis in Polish language

Wykorzystanie drzew zależnościowych w automatycznej odmianie złożonych wyrażeń w języku polskim

Contents

1 Introduction

Natural Language Processing (**NLP**) is a burgeoning field lying on the intersection of linguistics, traditional computer science, and artificial intelligence. The first forays into this domain can be traced back to Chomskyan revolution. These efforts largely followed what can be called the 'classical' paradigm of NLP: they were based on formal models of natural languages, and algorithmic methods of manipulating data. Over time, a more quantity-oriented approach emerged, which used statistical methods, in order to construct more flexible means of handling language data. This latter 'statistical' approach gained much traction, with the growing popularity of machine-learning in the previous decade.

An important disadvantage of the statistical paradigm, consists in its lack of explanatory power. Machine learning models, such as neural networks, are often just ensembles of equations with billions of numerical parameters, which after an arduous process of training happen to converge on producing the right answer with satisfying frequency. The ways of interpreting why, and how, these models happen to acquire capabilities of performing complex linguistic tasks, are very limited[1]. However these are mostly omissions with respect to theoretical virtues, while most works in the field are driven by practical concerns, and so the criteria for evaluating them are more pragmatic in nature.

In many cases the applications of NLP techniques are very specific, and therefore so are the criteria of evaluation, and the theoretical significance of researching these problems can be marginal. So is the case perhaps in the domain of Natural Language Generation (**NLG**). In general, the task of producing creative, relevant, and both syntactically and semantically correct language may seem to have even philosophical implications. But in many real-world applications, creativity is of little importance, and can even be a hindrance. A black-box, uninterpretable model, can also be unpredictable (this, one might argue, being the crux of its creativity), which can, and has, lead to negative outcomes. A spectacular example comes from Microsoft's Tay fiasco in 2016. A chatbot which learned in a continuous feedback loop of interactions with people via *Twitter*, was hijacked within 24 hours of going public, and turned into a prolific disseminator of rather unparliamentary

[1]Although there is a growing interest in ways of making these models interpretable, which will likely be necessary, for widespread adoption in everyday life.

opinions. A more mundane scenario, of natural language generation going awry, may occur in any interaction, in which a misleading message produced by a system may lead to people making costly decisions, e.g. in the financial, or medical domain.

Designing interfaces (i.e. mechanisms for efficient communication between agents, such as humans and machines) is a field of work, in which pragmatic concerns reign supreme over theoretical ones. For many of our interactions with the world (e.g. ones depending on precise spatial dimensions) visual or tactile modalities are inescapable. But there are many other cases, where it simply is more convenient to do things with our words, as opposed to with our hands. This is because the structures inherent in language (e.g. quantifiers) can be more attuned to these tasks (e.g. querying databases) than purely sensory ways of organizing data. A mature Natural Language User Interface (**NLUI**), capable of substituting other means of interfacing (e.g. buttons, menus, lists and forms, used in GUIs and web interfaces), will inevtiably have to involve robust NLG capabilities.

1.1 Motivation

This thesis is concerned with a particular method (using dependency grammar) of automatic inflection, which extends it to a particular application (complex phrases). Its ultimate purpose is to aid in Natural Language Generation for a particular language (or class of languages). It is therefore desirable, that applying this method in NLG systems, will lead to improving their capabilites. It has been argued above that this can be done irrespectively of the theoretical significance of this method, e.g. whether it mimics in any way language generation in humans, or whether the linguistic structures used in it, come from a true theory of grammar. In general, the criteria for evaluation of an NLG system can be enumerated as follows:

1. **Semantical correctness**: the messages generated by the system should express the desired intents well.

2. **Syntactical correctness**: the messages generated by the system should satisfy syntactical criteria of well-formedness of the language.

3. **Flexibility**: the system should be able to express the full breadth of intents needed by its application.

4. **Control**: the system should be predictable, and stable, to prevent possible risks associated with its application.

None of these criteria are easy to quantify, and they might even involve some trade-offs. For example increasing **flexibility** will tend to reduce the **controlability** of the system, as the number of possible intents to express grows. It will be argued, that the solutions provided as part of this thesis, will increase the flexibility of NLG applications, while not damaging any of the other criteria.

1.2 Template-based NLG in English

As with other problems in the domain of NLP, research into NLG has so far been concerned mostly with English. The first notable example of such a system is *Eliza*[15], the chatbot-therapist. Eliza was in essence just a clever trick, but she managed to fool her contemporaries into thinking that they were talking to a genuine human being. The Natural Language Understanding (**NLU**) and NLG were intertwined as two sides of the same mechanism — transformation rules. These rules converted input messages into output by means of extracting certain phrases, and then pasting them into predefined templates. The means of extraction were syntactical in nature: by exploiting rigid word order of English, Eliza was able to, for example, extract the predicate, and then use it in her own response. Consider the following rule:

Upon receiving a message of the form: **0** YOU **1** ME

Reply with: WHAT MAKES YOU THINK I **1** YOU

The numbers in the templates signify slots to be filled by expressions of arbitrary length. In the example above, slot **1** will be filled with whatever words occur between '*you*' and '*me*'. Because English has **SVO** order, in most cases we may reason as follows. The subject role can be assigned to '*you*' and the object role to '*me*', and whatever words lie between these, have to fill the verb (predicate) role, which expresses a relation, in which '*you*' (Eliza) is the active, and '*me*' (the interlocutor) is the passive part. This is enough semantic information, to ensure that the reply, generated by extracting whatever fell into slot **1**, and putting it in the response template,

will not be out of place. For a more concrete example, providing the following as input:

It seems to me that you don't respect me.

will provoke the response:

What makes you think I ***don't respect*** *you?*

The text produced is semantically relevant, and syntactically correct. Template based approaches worked well for many English applications, because phrases could simply be inserted into sentences, like building blocks, as long as they are put in a place that fits their grammatical role.

1.3 Template-based NLG in Polish

The success of simple template-based NLG in English is rooted in the relative simplicity of morphological structure of the language. But this is not true, of many other languages, in which a relaxed word order is balanced by rich morphological structure. In such languages, there is often no one proper place to insert a word into a sentence, even if it is known in advance which grammatical role it is supposed to fill. Instead, the grammatical role is partially reflected in the morphology of the word, and so the word must be inflected for the sentence to be well formed. In these cases words can not simply be taken from one context, and inserted into gaps in another, as this might produce ungrammatical sentences. For instance, the nominative case is often associated with the subject role, whereas accusative case corresponds to the object of the sentence. A template based approach would have to respect these constraints.

All of this is true of Polish, with its comparatively rich case system, grammatical gender, and conjugation paradigms[2]. Consider attempting to capture the same transition for Polish:

Wydaje mi się, że [Ty] mną gardzisz.

Dlaczego myślisz, że Tobą ***gardzę****?*

[2]As a rough measure of the scale of the problem, the biggest morphological dictionary for Polish — sgjp.pl — has 7.55 orthographically different forms for each nominal lexeme, and 26.47 forms for each verbal lexeme.

Apart from difficulties in capturing all the possible word orders, which amount to the same meaning (cf. '*Wydaje mi się, że gardzisz mną.*'), what is expressed by a distinct pronoun form in English ('*You*'), in Polish can be expressed by just conjugating the verb ('*gardzić*' → '*gardzisz*'), making the pronoun redundant. This makes classifying the input harder, but even if that is achieved, and the verb phrase is extracted, inserting it into the template as is, would produce ungrammatical text.

Dlaczego myślisz, że Tobą* ***gardzisz**?*

It is common for Polish speakers, to interact with interfaces which do not take morphology into account. The messages generated in this way sound artificial, and are often taken as signaling that one is interacting with a computer system instead of a real person. For this reason, designing seamless natural language interfaces, such as dialogue agents requires a morphology-sensitive framework.

Nevertheless it is a mistake to think, that this is just a matter of avoiding linguistic awkwardness. Grammatical systems and rules each have their purpose in communication, and so is the case with the case system, and grammatical gender (e.g. for anaphora resolution) which are present in Polish. Being able to satisfy the constraints imposed on language by these systems, is crucial if NLUIs are to become more than a technological novelty.

1.4 Extension into complex phrases

Suppose there was a method of solving the issues outlined above. First of all, that there was a sufficiently robust system for NLU to tackle issues of word order, and recognizing which words should be extracted. Second of all, that given these extracted words, we knew what was the desired form of the word, and had the means of inflecting the word into that form. So far this does not translate into an ability to work with larger phrases as opposed to individual words. And yet there is nothing about the templates above, which would limit the complexity of the structure of whatever is put inside the gap. Consider, as an example, a nominal phrase:

Biały nóż od Wojtka

which is to be inserted into the following template:

Piotr wystraszył go ___

The verb '*wystraszyć*' requires that its indirect object be in the *instrumental* case, so ignoring inflection altogether will leave us with the ungrammatical:

**Piotr wystraszył go biały nóż od Wojtka*

Inflecting the words is necessary, but treating them independently will also produce an ungrammatical result:

**Piotr wystraszył go białym nożem od Wojtkiem*

What is instead necessary, is to inflect the phrase as a whole, taking into account its grammatical structure. Phrases are, by definition, not mere assemblages of words, but instead exhibit internal syntactical structure. In Polish, these structures are reflected in morphological features especially, and so a problem arises, which is entirely alien to Enlgish NLP. Only by achieving a solution to this problem, can we obtain the grammatically correct version:

Piotr wystraszył go białym nożem od Wojtka

1.5 The aim of this work

This thesis is concerend with describing a method for accurately inflecting Polish phrases, for the purposes of template based generation. The aim is to implement a procedure, which will respect syntactical constraints, and be flexible with respect to possible applications. The implementation will be embedded within standard contemporary NLP setting: a component to a model in a popular Python library[3].

2 Theoretical framework

In section 1 it has been argued, that in many works in the domain of NLP, theoretical considerations are secondary to pragmatic concerns. Regardless, it will be argued here, that in the problem at hand, certain theoretical tools will be of great use. It should however be observed, that these theories are of use here, in virtue of being embodied in particular resources and tools for working with Polish language, as opposed to their more abstract characteristics.

[3]The code for the implementation is available at `https://github.com/ryszardtuora/flexer_thesis`.

It has already been remarked above, that using complex phrases in template-based NLG, should take account of their syntactical structure. It will now be fleshed out what sort of structures are of particular importance here, and how these can be processed computationally.

The traditional ('received-view') account of Polish grammar, mentions two types of syntactic relations:

- **agreement** — two grammatically related words are made to exhibit the same value of a certain grammatical attribute.
- **governance** — one word forces some grammatical phenomenon on another, e.g. inflection of its argument into a particular form, or introduction of an auxillary word.

In the standard contemporary account of Polish grammar [12] both of these are subsumed into one category of *syntactical accommodation* — understood as an assymetric relation, defined, for the sake of generality, between *phrases*, as opposed to *individual words*. The concept of syntactical accommodation is then divided into morphological accommodation, and nonmorphological accommodation, the latter collapsing further into lexical accommodation[4], and purely syntactical nonmorphological accommodation[5].

In this particular application, the most important constraint to take into account is, is morphological accommodation. In template-based NLG, the phrases to be inserted usually fill the role of arguments of their embedding constructions. If these embedding constructions impose any lexical requirements, these are usually with respect to morphologically invariant words, such as prepositions, and so these can just be incorporated into the templates statically, before any dynamic generation. The same argument applies to lexical accommodation within the phrases to be inflected. Provided the phrases are well formed, it can be assumed, that all lexical constraints have been satisfied. On the other hand, the concept of purely syntactical nonmorphological accommodation applies to rather rare, and subtle phenomena, which are usually outside the scope of NLG.

A mechanism for working with morphological relations will have to interface with two types of information. First of all it will have to work with morphological features of individual words, and so it will have to be able to represent these *via* computational formalisms for morphology. Second of all,

[4]E.g. obligatory introduction of particular prepositions by certain verbs.

[5]E.g. requirement of a particular structure of a phrase: forcing agglutination of the '*–ś*' form to a pronoun, or requiring clauses as opposed to nominal phrases

for the manipulation of morphological features to be sensitive to syntactical structure, an interface with a grammatical formalism will be needed. These two subjects will be discussed in the following two subsections.

2.1 Morphological framework for Polish

The most important criterion for choosing a theoretical framework for a project in NLP, is the availability of tools and resources. [12] was the theoretical background against which the biggest resource for Polish — The NKJP[11][6] corpus — was created. The tagset devised for description of Polish morphology used in this corpus, was subsequently developed to form the basis of the most popular tool for morphological analysis in Polish: **Morfeusz 2**[6], and the associated SGJP[7] dictionary. A more recent formulation of this framework is [17]. All these resources are considered standard by NLP practicioners in Polish, and so they are a natural choice for this project as well.

This tagset (henceforth referred to as the **NKJP tagset**) is a set of grammatical categories, each paired with a set of their possible values. Each form can then be associated with a mapping from the set of grammatical categories to the particular values it exemplifies (its morphological *tag*, or *profile*). This mapping will always be incomplete, i.e. there is no word in Polish, which would exhibit a value for all categories. Usually the grammatical class of the word, determines which categories it will exemplify (e.g. nouns can be inflected with respect to case, and number, but have a fixed gender and do not exhibit mood or degree). Because the names of the values are unique, there is no ambiguity in omitting the category names, and just listing the values, separated with a separator(":"). As these features are usually ordered according to a predefined order, it is often described as a positonal tagset. Therefore a tag in the NKJP tagset might look like this:

'*jasny*' — ADJ:SG:ACC:M3:POS

The first position specifies the grammatical class (here, an adjective), and the subsequent ones define its number, case, gender, and degree. The full tagset, with examples of values for each category is specified in Table 1.

[6] `http://nkjp.pl/`
[7] `http://sgjp.pl/`

Number: (2 values)		
singular	sg	*oko*
plural	pl	*oczy*
Case: (7 values)		
nominative	nom	*woda*
genitive	gen	*wody*
dative	dat	*wodzie*
accusative	acc	*wodę*
instrumental	inst	*wodą*
locative	loc	*wodzie*
vocative	voc	*wodo*
Gender: (5 values)		
human masculine (virile)	m1	*papież, kto, wujostwo*
animate masculine	m2	*baranek, walc, babsztyl*
inanimate masculine	m3	*stół*
feminine	f	*stuła*
neuter	n	*dziecko, okno, co, skrzypce, spodnie*
Collectivity[8]: (2 values)		
collective	col	*dwoje, dzieci*
noncollective	ncol	*dwa, okna*
plurale tantum	pt	*drzwi, wymiociny*
Person: (3 values)		
first	pri	*bredzę, my*
second	sec	*bredzisz, wy*
third	ter	*bredzi, oni*
Degree: (3 values)		
positive	pos	*cudny*
comparative	com	*cudniejszy*
superlative	sup	*najcudniejszy*
Aspect: (2 values)		
imperfective	imperf	*iść*
perfective	perf	*zajść*
Negation: (2 values)		
affirmative	aff	*pisanie, czytanego*
negative	neg	*niepisanie, nieczytanego*
Accentability: (2 values)		
accented (strong)	akc	*jego, niego, tobie*
non-accented (weak)	nakc	*go, -ń, ci*
Post-prepositionality: (2 values)		
post-prepositional	praep	*niego, -ń*
non-post-prepositional	npraep	*jego, go*
Accommodability: (2 values)		
agreeing	congr	*dwaj, pięcioma*
governing	rec	*dwóch, dwu, pięciorgiem*
Agglutination: (2 values)		
non-agglutinative	nagl	*niósł*
agglutinative	agl	*niosł-*
Vocalicity: (2 values)		
vocalic	wok	*-em*
non-vocalic	nwok	*-m*
Fullstoppedness: (2 values)		
with full stop	pun	*tzn*
without full stop	npun	*wg*

Table 1: The set of all grammatical categories, and their possible values, adapted based on `http://nkjp.pl/poliqarp/help/ense2.html` and [7]

2.2 Dependency grammar

With respect to syntactic analysis, there is a wide range of linguistic formalisms. Initially, the constituency grammar paradigm, championed by Chomsky was more popular. This approach analyses a sentence as recurrently composed of constituent phrases. It has already been remarked above, that [12] favoured defining syntactical accommodation as occuring between phrases, as opposed to individual words. This is because of the examples like the following:

Zagniewani Ania i Maciej poszli

both the adjective '*zagniewani*' and the verb '*poszli*' occur in the plural number, despite the nouns '*Ania*' and '*Maciej*' appearing in the singular form. This is because they enter into the grammatical relation of accommodation with other words in the example only after forming the coordination '*Ania i Maciej*' which can be said, to exhibit the plural number.

However, presently, mainly because of technical reasons, a different annotation scheme is far more popular in NLP — dependency grammar. This paradigm analyses a sentence as a tree, the nodes of which are individual words, connected by arcs (dependency relations). The arcs can be labeled, to distinguish different types of relations, e.g. that an adjective modifies a noun, or that a noun is the subject of a verb. Both the exact guidelines for drawing arcs (which words are connected with each other, and which way the arc goes) and the set of labels, are the subject of a scientific discussion. At the moment the most well established set of guidelines, is the Universal Dependencies (**UD**) methodology[9]. UD is an interlinguistic project, aiming to capture the syntax of all languages in one framework of 37 types of relations. The topology of the syntactic trees is constructed according to a broadly 'semantic' methodology: for a pair of connected words, the one which more closely resembles the semantics of the pair taken as a phrase, is chosen as the head. This means, among other things, that function words will generally be subordinated to content words they come in relations with[10].

[8] *Przyrodzaj* is the working name for this category (per [7]), there is no established translation as of yet. It limits which form of a numeral, a noun can be associated with in constructions.

[9] `https://universaldependencies.org/`

[10] An opposite methodology is employed in **SUD** `https://surfacesyntacticud.github.io/`.

The approach chosen here, is to approximate the representation of accommodation with the use of syntactical analyses provided by dependency grammar in the UD annotation scheme (in particular, in the vein of the largest UD treebank for Polish — **PDB**[18]). The main reason for this, is accessibility of guidelines documentation, corpus data, and also the predominance of dependency grammar oriented parsers in the NLP toolset. This is true for Polish, but also for many other languages, which means that the algorithms described below, can also be applied to other languages, as long as a similar account of morphology will be provided. Some dependency arcs will be taken to "carry" morphological accommodation with respect to particular attributes, based on their type. For example the AMOD (adjectival modifier) relation, will usually involve accommodation of gender, case, and number between the head and the dependent word.

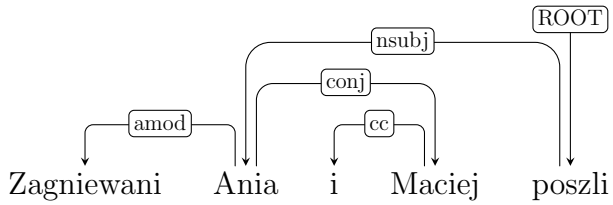

Figure 1: A dependency analysis of the example above, according to the UD guidelines. Note that in UD the coordination is headed by one of its conjuncts, as opposed to the '*i*' conjunction.

To a certain extent, dependency analyses can offer similar information to constituent ones. Constituency phrases usually have a central word, with which they share many of their features. In many cases the subtree spanned by the word in the dependency analysis of the same sentence will overlap with the constituency phrase. But the emergent features, such as the plural number in the example above, is not something that can be captured in a dependency analysis. Therefore, using the dependency grammar as a formalism for representing syntax, should not be taken as a serious proposition in linguistics, but rather as a pragmatic proxy for the actual accommodation relations, to be used in an NLP application.

2.3 Scope of interest

Given the theoretical introduction, an outline of linguistic phenonomena which will be captured in the final implementation can be stated:

- Nominal phrases: *Biała flaga z papieru*
- Phrases with numerals: *Czworo małych dzieci*
- Adjectival phrases: *Pochodzące z Lazurowego Wybrzeża*
- Coordinations: *Skórzany but i jeansowe spodnie*
- Proper names: *Uniwersytet Papieski Jana Pawła II w Krakowie*

These types of phrases account for the majority of applications in template-based NLG. It should be noted, that this excludes most verbal constructions. Because these usually form the centers of sentences, they also tend to form the backbone of templates, with their arguments being left to be filled as empty slots. Therefore such a limitation is not very troublesome in standard NLG solutions. Exclusion of these constructions is an important simplification, because in Polish they tend to involve the so called 'analytic mode' of inflection, which involves using auxiliary words, or agglutinants. The assumption that underlies this work, is that inflection of a phrase does not alter the number of segments within the phrase. This is mainly because adding mechanisms for adding or subtracting words, would have to be very language specific, and language-agnosticity is one of the desiderata for this system.

3 Implementation

Implementing the ideas discussed above requires embedding the inflection functionalities in an environment, in which information about the morpology and the grammatical structure of phrases is available. Moreover, these features have to be specified in an annotation scheme, which follows the assumptions made above, i.e. the morphological information has to conform to the NKJP tagset, and the grammatical structure has to follow the guidelines of the UD dependency grammar. Because the availability of tools was one of

the main reasons for choosing the formalisms, a ready-made pipeline is available in the form of the *pl_nask* model[11] to the popular NLP library: **spaCy**. This model includes three components of interest:

- **tagger** trained on NKJP, which assigns a morphological tag to each form in the text.
- **lemmatizer** based on a dictionary, which pairs each form in the text with its lemma — a representative form of the lexeme it belongs to.
- **dependency parser** trained on PDB, which assigns a dependency head and relation label to each token in a text (this is also used for segmenting text into sentences).

These components will be used to achieve a representation of the morphology and structure of the phrase which is to be inflected, and put into a template.

3.1 Task description

A distinction is often made between **inflection**: the task of finding a form of the lexeme which satisfies the desired morphological profile, given its lemma, and **reinflection**: where the base form can be any form belonging to the lexeme. If the morphological profile is incomplete (i.e. it is satisfied by more than one form of the lexeme), reinflection will also be taken to involve completion of the profile in accordance with the base form, i.e. finding the form which satisfies the morphological profile *and* is closest to the base form (e.g. by preserving its grammatical case). Since the final task, is inflecting phrases, which possess internal structure (and so in most cases are not composed purely of forms identical to their lemmas), it will involve reinflection, but we will not observe this terminological distinction strictly. This is because given a reliable tagger and lemmatizer, reinflection can be reduced to inflection of the lemmatized form, with provisions based on the current tag.

The basic case of the task is reinflecting a single word. Given a word $\boldsymbol{w}$ (not necessarily identical to its lemma $\boldsymbol{l}$), with a particular morphological profile $\boldsymbol{p}$ — a set of morphological features — and the target set of morphological features $\boldsymbol{t}$, the goal is to find a form of the same lemma $\boldsymbol{w*}$ which satisfies all the features from $\boldsymbol{t}$ *and* is associated with the morphologically closest profile $\boldsymbol{p*}$. It should be noted that in many cases $\boldsymbol{t}$ will not include

[11] http://mozart.ipipan.waw.pl/ rtuora/spacy/

values for all the morphological attributes, which means, that remaining features in $\boldsymbol{p}$ should be left unaltered[12]. Such is the case e.g. when it is our goal to obtain a plural form of a word, without altering its grammatical case.

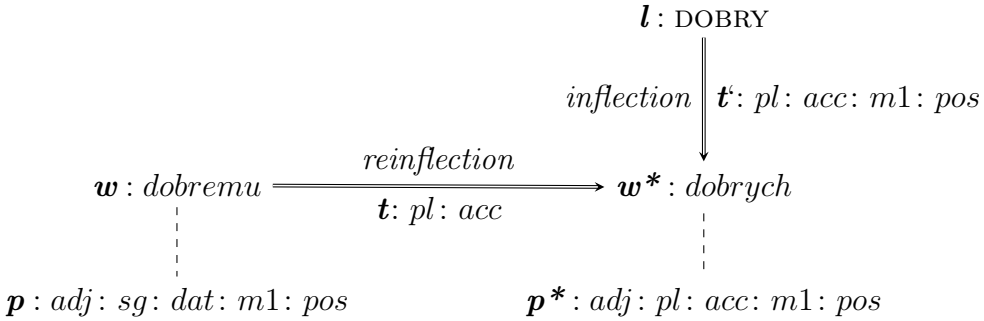

Figure 2: Inflecting '*dobry*', and reinflecting '*dobremu*' into '*dobrych*'. The degree and gender are carried over from the original morphological profile.

3.2 Single word inflection

In what follows, two methods for inflecting individual words will be described: dictionary based inflection, and a neural network inflection mechanism. They can be considered to be functionally similar, as they fulfill the same role in the broader system, but their use cases will be quite disparate, as they differ substantially in their capabilities.

3.2.1 Dictionary based inflection

Inflection of a single word can be done by using an inflection dictionary as follows. First the set of all forms associated with the lemma[13] of the word $\boldsymbol{w}$ is recovered, and filtered down to forms which satisfy all the features from target profile $\boldsymbol{t}$ (see Table 2). Secondly, the list of these forms is sorted by the size of the symmetric set difference defined on morphological profiles of the base form[14], and each of the candidate forms. Lastly, the top rated word is

[12]And this is the sense of proximity, implied in "morphologically closest". It should be observed that sometimes this is not possible, because certain feature combinations might be illegal. This is why a distance metric, as opposed to a hard constraint of satisfying remaining features will be used below.

[13]This lemma is available thanks to the lemmatizer.

[14]Which is available thanks to the tagger.

returned as the form which satisfies the desired morphological profile, and is the closest to the original form. In this implementation Morfeusz 2 [16] was used for interfacing with the SGJP dictionary.

	m1	others
pos	**dobrych**:4	dobre:6
com	lepszych:6	lepsze:8
sup	najlepszych:6	najlepsze:8

Table 2: The set of all candidate forms belonging to the lexeme for "dobremu": the adjective DOBRY (*good*), which has been narrowed down to forms satisfying the target morphology t (*pl:acc*). This leaves two attribute values to still be selected: degree (rows) and gender (columns). The morphologically closest form (in bold) is chosen based on the size of the symmetric difference between profiles (given after the colon), in this case the profile closest to the original ADJ:SG:DAT:M1:POS is in M1 gender and POS degree:

In listing 1 one can see two functions, adapted from their original implementation as methods of a `Flexer` class object. `tag_to_feats` converts a tag to a set of values for grammatical attributes. It can optionally enrich the tags with default values for some attributes (the purpose of this functionality will be explained in subsection 5.2.1). This function is used for converting from original positional tag string representations, to sets of strings representing features. `dict_flex` is then used to list all the possible realizations of the lexeme associated with the `lemma` argument, and subsequently that list is narrowed down to forms associated with feature sets being supersets of the target feature set. Afterwards this list is ordered according to the numerosity of symmetrical difference with the original tag, in order to find the morphologically closest form. The first and second arguments to the `dict_flex` function, are provided (in the context of the full application) by the lemmatizer and tagger respectively, based on the user input, the third is given by the user themselves. As a result, inputting `"dobremu", "adj:dat:sg:m1", "pl:acc"` as arguments will yield the desired output: '*dobremu*'.

3.2.2 Neural inflection

A *sequence-to-sequence* neural network will now be described, that attempts to capture the same capability of inflecting individual words. A dictionary

```python
def tag_to_feats(tag_string, keep_pos=False, enrich=False):
    split_tag = tag_string.split(":")
    if enrich:
        # finding unfilled attributes and assigning default values to them, based on the dict objects representing the tagset
        filled_attrs = [VAL2ATTR[val] for val in split_tag[1:]]
        unfilled_attrs = [attr for attr in ATTR2FEATS if attr not in filled_attrs]
        split_tag = split_tag + [DEFAULT_VALS[attr] for attr in unfilled_attrs]
    if not keep_pos:
        split_tag = split_tag[1:]
    return set(split_tag)

def dict_flex(lemma, current_tag, target_feats):
    target_feat_set = set(target_feats.split(":"))
    lexeme = morf.generate(lemma) # generating all forms of the lexeme using Morfeusz2
    current_feats = tag_to_feats(current_tag, keep_pos=True,
                                 enrich=True)
    candidates = []
    for form in lexeme:
        form["feats"] = tag_to_feats(form["full_tag"],
                                     keep_pos=True,
                                     enrich=True)
        if target_feat_set.issubset(form["feats"]):
            form["score"] =
             len(current_feats.symmetric_difference(form["feats"]))
            candidates.append(form)
    if not candidates:
        return None # no satisfactory form found
    ranking = sorted(candidates, key=lambda form:form["score"])
    inflected_form = ranking[0]["form"]
    return inflected_form

dict_flex("dobremu", "adj:dat:sg:m1", "pl:acc") # -> dobrych
```

Listing 1: The dictionary based inflection functions.

based solution will generally be superior, as it uses a resource manually curated with respect to linguistic correctness. Nevertheless it is completely unusable for working with lexemes which are outside the dictionary. This encompasses two classes of words: neologisms (which have not yet been recognized by the lexicographers), and domain- and idiolect-specific vocabulary (which is outside the scope of general purpose lexicons). These linguistic phenomena, although often exhibiting non-standard phonological and orthograpical mechanisms (e.g. loanwords or humorous word formation), are in most cases quite regular in their inflection patterns. This is important because a generative solution (such as a neural network) will hardly be able to infer irregular inflection patterns, but generally has no explicit limitations with respect to the input it can handle. Generative methods of automatic inflection have been the subject of extensive study in the previous years[15], but they do not surpass a simple, dictionary based system, for the most common words. For these reasons a neural inflection method is mainly reccomended for words which are outside the dictionary.

A sequence-to-sequence (or *seq2seq*) neural network is a type of a recurrent neural network, which can take in, and output sequences of arbitrary length. It is composed of two components: an encoder, and a decoder. Normally the encoder is a recurrent neural network, which passes through all the timesteps of the input sequence, and condenses its representation into a single fixed length vector. Then a decoder takes in that vector as input, and produces steps of the output sequence by classifying over a fixed set of symbols until a special END token will be produced. In this application though, the decoder will not use a single vector representation of the input, but rather an *attention* mechanism, which is able to selectively aggregate information from all the timesteps of the input sequence. All the hidden states of the encoder (as opposed to just the last one) are used, and passed in to a separate layer of the network, which outputs weights for each timestep. The final output of the attention layer, is a weighted average of all hidden states of the encoder.

[15]Cf. the SIGMORPHON shared tasks [1], [2] which have dealt with automatic inflection across a broad spectrum of languages.

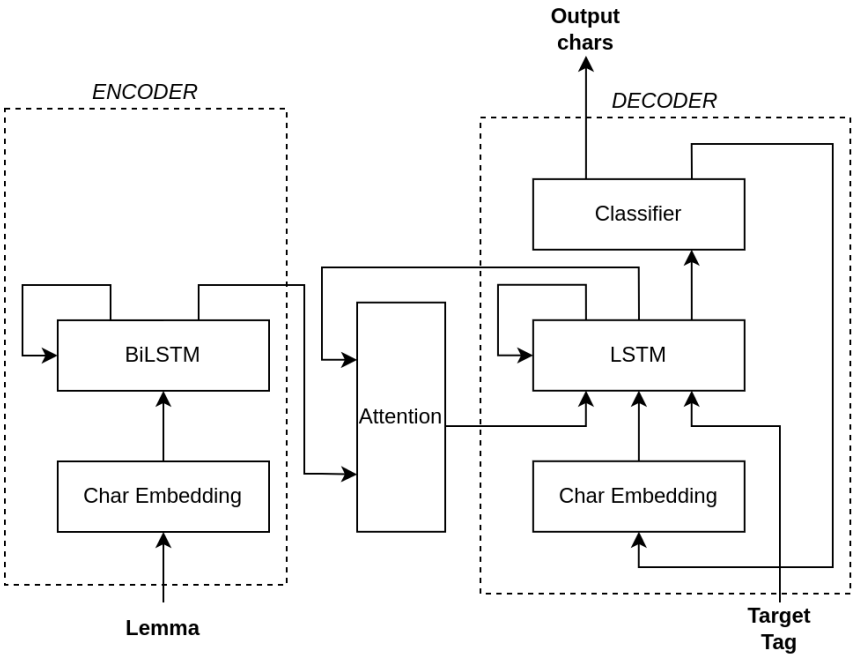

Figure 3: Architecture of the neural network. It was loosely based on [5]

Attention[16] is important because it allows to represent a nontrivial correspondence between characters in the input sequence, and characters in the output sequence. Inflection in Polish is dominated by suffixes, with some prefixes (e.g. the negation '*nie-*', or the superlative prefix '*naj*'). In a typical case most of the characters just have to be rewritten from the lemma, with some modifications to the ending. In the case of prefixes however, the mapping has to be shifted, and it is important for the network, to be able to learn these mappings. Another important mechanism in Polish is apophony (e.g. '*noga*' → '*nodze*'). The network has to learn these phonological patterns in order to be able to inflect words efficiently.

			p	i	ę	k	n	y				
			‖	‖	‖	‖	|					
n	a	j	p	i	ę	k	n	i	e	j	s	z	y

Figure 4: Character correspondence between '*piękny*' and '*najpiękniejszy*'

[16]In this case in particular: an additive soft attention layer.

The inputs to the network are the one-hot encoded representations of each lowercased character in the word, and the full desired morphological tag. For the purposes of training the model, a morphological inflection dictionary can be used as a dataset. All the 7.5 million (FORM, LEMMA, TAG) triples are extracted from the SGJP dictionary. FORM is the gold output expected from the network, whereas LEMMA and TAG are passed in as input[17]. This is an immense dataset, but it is very imbalanced, as the number of training examples per a particular inflection pattern does not in any way reflect its importance. Zipf's law states, that a word's use frequency is inversely proportional to its place in the frequency list. This entails, that the most frequent words vastly outnumber less popular ones in an average text, and yet in this dataset they are represented on par, with words of negligible frequency. Moreover, the most popular words, are much more likely to exhibit unique, and nonregular patterns of inflection. For example in Table 3, a paradigm for the second person personal pronoun, a very common word, is shown. It it is immediately clear, that there is no single character which stays fixed in all of the forms, and there is a large degree of syncretism with respect to both cases, and accentability.

	sg	
	akc	**nakc**
nom	ty	
gen	ciebie	cię
dat	tobie	ci
acc	ciebie	cię
inst	tobą	
loc	tobie	
voc	ty	

Table 3: Inflection paradigm for the second person pronoun '*ty*'.

For this reason, a countermeasure was introduced, which aims to mirror the actual frequency of words. A frequency list calculated on the 300 million version of the NKJP corpus was used, to model the probability distribution of lexemes. Instead of using the dataset in its entirety (which would

[17]This particular choice of training data is also the reason why the neural inflection system can not outperform the dictionary based method.

be prohibitively expensive computiationally), forms for each batch are sampled from the the dictionary, with the probability based on the frequency of their lemmas in the corpus. This however introduces another difficulty, as many of the most frequent words are also function words such as the coordinating conjunction '*i*' or the preposition '*do*', and these tend to be morphologically invariant. For this reason, words which belong to invariant grammatical classes are penalized by having their frequency divided by 100.

The model has been trained for $150k$ batches, 256 examples each. The final accuracy on the test set was 76.71%. In listing 2 some examples of the behavior of the system are shown. It is notable, that the system has no problem with inflecting neologisms, but still breaks down on common, yet irregular inflection patterns. It also appears that it has a preference for longer words, as they might provide more characters to make it clear, which inflection paradigm is most relevant in this case. In listing 2 some examples of the performance of the network are shown. Neither the neologisms[18], nor the domain vocabulary[19] is present in the SGJP dictionary, but because these words follow regular and ordinary inflection patterns, the system is able to handle them successfully. Some capabilities with respect to basic derivation mechanisms (as specified in subsection 5.2.1) are also demonstrated. The failures of the system often exhibit features of repetition[20], e.g. '*l-że-że-ksze*', sometimes the network is able to memorize the stem change, but is unable to apply the suffix properly, e.g. '*ludzi-ekom*'. These limitations confine the neural inflection system, to the areas of application where a dictionary can not suffice.

3.3 Inflecting phrases

Because phrases can be treated as if they possessed morphological features of their own[21], the extension of the task can be specified equivalently. That is to say, that it might be said that the entire phrase $\boldsymbol{w}$ exhibits a particular morphological profile $\boldsymbol{p}$, and one might want to inflect it into another profile $\boldsymbol{p*}$.

[18] Taken mainly from the yearly '*Młodzieżowe słowo roku*' contest.

[19] Taken mainly from the automotive domain.

[20] This is actually a common feature of recurrent neural networks, as discussed in [14].

[21] And, as observed above, sometimes these features are irreducible to the features of their components.

```
# Domain vocabulary
"pompowtryskiwacz", "subst:pl:dat:m3" # 'pompowtryskiwaczom'
"przedwtrysk", "subst:acc:pl:m3" # 'przedwtryski'
"rozdzielaczowy", "adj:sg:dat:pos" # 'rozdzielaczowemu'
"szkiełkować", "praet:pl:f:imperf" # 'szkiełkowały'
"izentropowy", "adj:pl:loc:com" # 'izentropowszych'
"przegrzew", "subst:sg:voc:m3" # 'przegrzewie'

# Neologisms
"zodiakara", "subst:pl:dat:f" # 'zodiakarom'
"śpiulkolot", "subst:sg:loc:m3" # 'śpiulkolocie'
"koroniak", "subst:m2:inst:pl" # 'koroniakami'
"tozależyzm", "subst:gen:pl:m3" # 'tozależyzmów'
"krindżowy", "adj:pl:m1:gen:sup" # 'najkrindżowszych'
"becelować", "pact:pl:m1:gen:neg" # 'niebecelujących'
"essa", "subst:pl:gen:f" # 'ess'
"kogiel", "subst:gen:sg:m3"# 'kogla'

# Idiosyncratic inflection patterns
"wiadro", "subst:sg:n:gen" # 'wiadr'
"lekki", "adj:sg:nom:m1:com" # 'lżeżeksze'
"człowiek", "subst:pl:dat:m1" # 'ludziekom'
"pies", "subst:sg:dat:m2" # 'psusowi
"mieć", "fin:pl:sec:imperf"# 'macie'

# Examples deemed challenging in the SIMORPHON 2021 task
"negatyw", "subst:sg:gen:m3"# 'negatywu'
"rabunek", "subst:sg:gen:m3"# 'rabunku'
"tekstylia", "subst:pl:gen:m3:pt"# 'tekstyliów'
"wiktuały", "subst:pl:gen:m3:pt"# 'wiktuałych'
"wyjaśnić", "impt"# 'wyjaśnij'
"podnieść", "fin:pl:ter"# 'podniosą'
"podejść", "praet:sg:sec:m1"# 'podszedłś'
"żagiel", "subst:gen:pl:m3"# 'żagli'
"szampan", "subst:sg:gen:m3"# 'szampanu'
```

Listing 2: Neural inflection of a choice of words. First the lemma is given, then the desired target pattern, the system output is listed as a comment. For the SIGMORPHON task see [10].

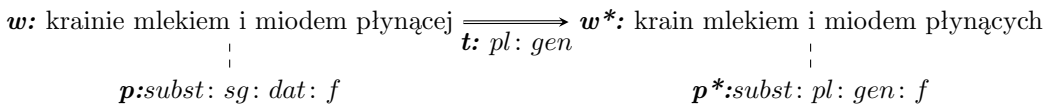

Figure 5: Example of a reinflection task for a complex phrase.

The subject of phrasal inflection is considerably less explored in literature, e.g. the interlinguistic SIGMORPHON task 2021 [10] contains multi-word inflection, but for Polish this is limited to the analytic mode of conjugation. The most detailed work for Polish is by [13], where a set of handwritten rules is used to tackle a very particular application of inflecting Polish toponyms. The technology used there is quite dated, and rules are specifically fitted to the domain of application, with no attempt at generalization. A more recent work for Czech template-based generation, is done by [4]. In their experiments, a neural generator is trained to produce sequences of lemmata, interspersed with morphological tags. The inflection of individual words is then done by reference to an inflection dictionary. These sequences also contain delexicalized "gap-tokens" which correspond to complex names, and they use a separate RNN to select proper surface-form based on the context.

3.3.1 Task decomposition using dependency trees

Given the assumption that the input phrase is well formed, a dependency tree of the phrase can be used to represent it.

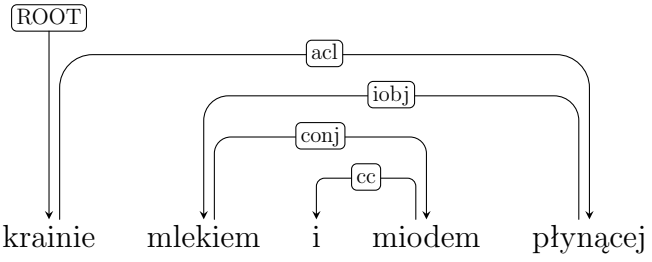

Figure 6: Dependency tree of the example above.

It is also assumed that the phrase forms a tree, or that it forms *one* subtree within its original context[22]. If that is not the case, the task will be addition-

[22]The dependency analysis has to be provided by the dependency parser.

ally decomposed into inflecting each of the independent subtrees individually.

In any case, the tree is rooted (or subrooted, if it comes from a bigger context, here this difference is irrelevant) at one of the tokens. This representation of the phrase, can then be used to recurrently decompose the task at hand. The base case, is that of inflecting individual words, and has already been discussed above. At each node in the tree, the task is then to inflect the word associated with that node (the base case), and then apply the same procedure to all the subtrees, spanned by each of its immediate children. For the analysis in Figure 6, inflection of the phrase, will involve first inflecting the root '*krainie*', and then inflecting the subtree spanned by its sole child: '*płynącej*' and so forth. In this context, it is crucial, that the solution for individual words is capable of tackling reinflection aswell, since keeping the unaccommodated features unaltered, is crucial to preserve other grammatical constraints.

The challenge is then to infer the precise parameters for each of the base cases, i.e. what morphological profiles to apply to each of the nodes. A trivial solution would be to apply the same morphological profile to all the nodes. Instead, as has been specified above, dependency relations will be utilized to represent accommodation relations. In more precise terms: some dependency relations will be taken to entail accommodation, and so any alteration made with respect to one of the accommodating features of the head, will be transferred to the child. In effect, the morphological features will be propagated throughout the dependency tree, at each arc being filtered by the dependency type. Because dependency structures contain individual words, and not phrases, the morphological features to be propagated, will be features of the words themselves. In particular, it will be heuristically assumed, that the morphological profile of the root will be used to represent the morphological profile of the phrase as a whole.

It should be remarked, that this procedure assumes, that inflection does not alter the number of words in the expression. This assumption is violated in some morphological mechanisms, such as reduplication, which is for example used to mark plural nouns in Indonesian. This is a flaw with respect to language-agnosticity of the system. In Polish, auxiliary words have to be introduced for inflecting many adjectives with respect to degree (e.g. the comparative form for '*cukrowy*' is '*bardziej cukrowy*', and in verbal paradigms (the so called analytic mode of conjugation), which are outside of the main scope of application of this system. Both of these mechanisms could be accommodated, using dedicated cases in the algorithm, but this would make

the algorithm itself language dependent in a structural way.

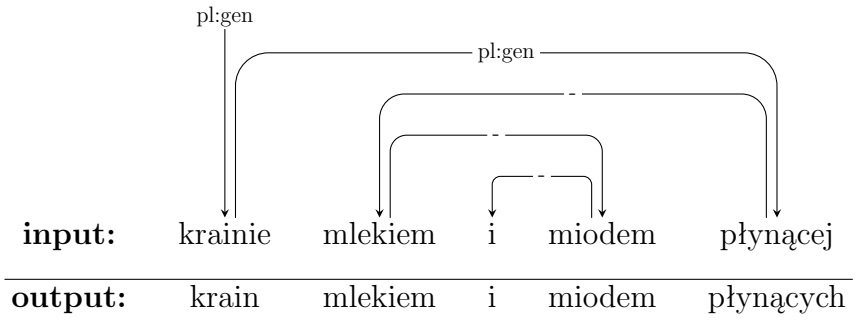

Figure 7: Propagation of features along arcs. '-' signifies that no feature should pass down this arc to the child, or any of its dependents. This is crucial not to disturb the existing constraints, e.g. the fact that '*płynących*' expects both '*mlekiem*' and '*miodem*' in the instrumental case, as opposed to genitive case passed in as the target case of the entire phrase.

The decision method for choosing which features to assign to each of the nodes could be modeled using a machine-learning approach, but a more traditional account, given in terms of rules, will be more customizable, and easy to understand and debug. Such a rule would have the form $dep \rightarrow attrs$ where *dep* is some description of the dependency relation, and *attr* represents the list of attributes, for which agreement[23] between the head, and dependent occurs. The simplest approach will be to represent dependency relations merely in terms of their dependency label in the annotation scheme, but more parameters could be added on, e.g. the grammatical classes of the head and subordinate words[24]. As an example: to model the grammatical fact that adjectival modifiers of nouns agree with them with respect to number, case

[23]The same formalism could be used, to model some government relations, but that would require a much bigger problem space, i.e. one would have to specify actual values for each attribute, as opposed to enforcing propagation.

[24]These additions would make the ruleset much more complicated, as it would require working with a product of features, as opposed to just one dimension. It would also reduce the amount of data per rule, which is undesirable. On the other hand the simpler rule format ensures, that the number of rules is equal to the number of dependency labels. Moreover expending parameters by POS tags is not really guaranteed to bring new information, because many dependency labels correlate strongly with particular grammatical classes.

and gender, a rule *amod* → *number:case:gender* can be introduced. This is the rule formalism that will be used in the rest of this work.

3.3.2 Rule induction

Although morphological agreement is a well studied phenomenon of the Polish language, it was not possible to find any existing rulesets formulated in terms of the chosen formalism. One might try to devise such a ruleset by hand, but it is an intricate, and error-prone process. Instead one might induce such a ruleset in an empirical fashion, i.e. by studying statistical regularities in a corpus containing both morphological, and dependency annotation. For each dependency label $\boldsymbol{l}$ all the arcs labeled with $\boldsymbol{l}$ are collected. For each attribute $\boldsymbol{a}$, the subset of arcs in $\boldsymbol{l}$, where both the head, and the child exhibit some value of $\boldsymbol{a}$ is then obtained. Finally the proportion of cases where the value of $\boldsymbol{a}$ is the same for both nodes of the arc is calculated. Because the aforementioned PDB treebank is also annotated with morphological information, it can be used for this purpose. Such a procedure assumes, that a high enough correlation of attribute values between dependents and heads, of a given dependency relation type, is good grounds for explaining it by reference to morphological agreement.

This procedure is clearly far from perfect, as there can be different systematic factors at play. For instance, it is often the case that nouns and their nominal modifiers agree with respect to number, but the reasons behind this are usually semantic, or pragmatic. Therefore, the procedure of rule induction, should be taken as a purely heuristic technique, which can be used to arrive at useful simplifications. As a result, we obtain a table of frequency of agreement given a dependency label of the relation, and the morphological attribute, like the one in Table 4. From this table, we extract all the agreement rules, by selecting those combinations which exceed a predefined threshold of **95%**.

	number	gender	case	person	collectivity
acl	**99.29**	**99.18**	**99.28**		
advcl	75.44	49.87	69.74	69.93	
amod	**99.78**	**99.81**	**99.48**		
amod:flat	**99.87**	**99.73**	**98.87**		
appos	94.33	82.57	88.30		90.91
conj	83.62	60.57	**95.97**	88.02	87.89
cop	**95.94**	79.75		80.00	
csubj	74.61	10.94	45.45	91.11	
det	**99.36**	**98.59**	**98.31**		
det:numgov	92.18	**97.67**	1.00		**100.00**
det:nummod	**100.00**	**98.91**	**100.00**		**100.00**
det:poss	**99.82**	**99.70**	**99.70**		
fixed	**98.22**	87.78	89.54		**100.00**
flat	**98.00**	92.92	92.77		**97.44**
nmod	67.20	26.46	23.24		78.13
nmod:arg	61.34	27.99	27.34		85.**99**
nmod:flat	81.85	55.66	65.78		88.89
nmod:poss	50.91	5.45	3.64		
nmod:pred	**100.00**				
nsubj	94.40	88.82	61.18	**98.51**	87.88
nsubj:pass	**98.27**	**97.59**	**99.13**		
nummod	**99.59**	**95.44**	**97.18**		**100.00**
nummod:flat	**100.00**	**100.00**	**100.00**		
nummod:gov	**96.97**	89.79	4.28		**98.67**
obj	62.85	20.61	40.59	36.59	**100.00**
obl	65.39	25.34	11.27	59.79	
xcomp	**95.48**	85.26	14.29	33.33	

Table 4: Frequency of agreement (in percent) for a selection of morphological attributes (columns), between the dependency head and its children, given the dependency label (rows). The empty cells indicate that no observations were found, for which the child is related to the parent via the dependency relation and both tokens exhibit some value of the attribute. The pairs which have passed the threshold are in bold. The data has been abridged from the full table obtained from analysing PDB treebank.

Most results are as expected, e.g. adjectival modifiers (*adj*) have very high ($\geqslant 99\%$) rate of agreement with respect to number, gender, and case. Similarly *det:poss* which corresponds mainly to possessive pronouns, scores very highly on the same attributes. On the other hand arguments such as direct objects (*obj*), or nominal modifiers (*nmod*) score significantly below the threshold on most attributes, as there is no syntactical basis for any substantial correlation. Nominal modifiers (*nmod*), score very low on grammatical case, as this attribute is fixed, either in the genitive, or some other case *via* a preposition. It should immediately be observed however, that some corrections need to be made by hand. For example:

- Appositions (*appos*) such as '*siostra Małgorzata*' would be expected to agree on both number (barely below the threshold) and case (much lower: 88.30%).

- Constructions marked with *flat*, mainly so called named entities (proper names, e.g. '*Andrzej Duda*', but also names such as '*drugi grudnia*' or '*ul. Jagiellońska 11*'), failed to exceed the threshold for case.

- Although not a matter of syntactical fact, it would be useful to be able to pluralize coordinations (e.g. '*pies i kot*' → '*psy i koty*'), and this would require introducing a corresponding rule[25]. The same argument can be made for gender, and degree (not included in Table 4).

- Nominal subjects (*nsubj*) do not exceed the threshold with their heads with respect to number, or gender. This is in contrast to the received view about accommodation, but the reason for this disparity, are the UD annotation guidelines themselves (mainly with respect to subjects constituted by numeral phrases, coordinations, and also some constructions involving copulae). Nevertheless, as this applies to constructions involving verbs, it is beyond the main scope of this work.

In what follows, the manual corrections described above, will be assumed to be part of the ruleset, e.g. the rule for *appos* will look like this: *appos* → *number:case*.

[25]To be more clear: there is no agreement with respect to number inside coordinations (otherwise constructions such as '*pies i koty*' would have to be ungrammatical). A more elegant solution might have been to have a dedicated mechanism to decomposing coordination structures, and inflecting each conjunct independently.

3.3.3 The algorithm for propagating features along dependency arcs

Listing 3 contains a function for recurrently inflecting subtrees. It invokes another function `filter_accommodable_feats` which interfaces directly with the ruleset and the representation of the tagset itself, in order to narrow down the features which will be propagated into subtrees. Based on the dependency label of each child's arc, the target feats are filtered by rules, and subsequantly passed to the recurrent call of the function. At each node, the paired token is inflected using `flex_token` which calls one of the two methods for inflecting individual words. The data is stored in a dict, which will be further processed to regain the initial ordering and formatting of the input.

```
1  def flex_subtree(head, target_feats):
2      ind_to_inflected = {} # the output is stored in a dict
       ↪  with token positions as keys
3      children_to_inflect = list(head.children) # using the
       ↪  dependency tree annotated by spaCy
4      inflected_head = flex_token(head, target_feats) # a
       ↪  wrapper for one of the single-word inflection methods
5      ind_to_inflected[head.i] = inflected_head
6      for child in children_to_inflect:
7          accommodable_feats = filter_accommodable_feats(child,
           ↪  target_feats) # filtering the features by the
           ↪  rules induced
8          inflected_subtree = flex_subtree(child,
           ↪  accommodable_feats) #recurrence
9          ind_to_inflected.update(inflected_subtree)
10     return ind_to_inflected
```

Listing 3: Recurrent inflection of phrases represented as subtrees.

3.3.4 Numeral phrases

The syntactical behavior of numerals in Polish (and their interaction with nouns in particular) is very subtle. In the NKJP tagset, this behavior is described by three attributes: case, collectivity, and accommodability. Most

nouns bind with noncollective (*ncol*) forms of numerals, but a select few (e.g. '*dziecko*') requires binding to collective forms (*col*). However there is no 1 : 1 correspondence between these features, as *plurale tantum* nouns such as '*skrzypce*', have a separate value of collectivity (*pt*), which, for the purposes of accommodation, is to be treated as equivalent to *col*. Additionally constructions with numerals can be either agreeing (*congr*) or governing (*rec*), the former accommodating on case of the noun, and the latter governing, imposing the genitive case on the associated nouns. This is problematic, beacuse in UD (in accordance with the semantic criterion for headedness) numerals are subordinated to their associated nouns. It is only after descending lower in the tree, that it may turn out, that a given noun (and its subtree) was not to be inflected, because it is bound by a governing numeral.

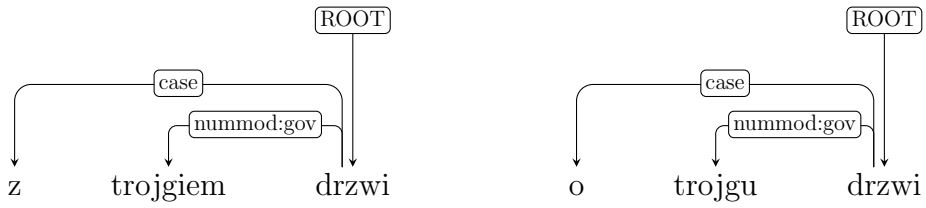

Figure 8: UD Dependency trees for two numeral phrases, in both cases it is the numeral that takes up the case imposed by the preposition, the noun stays in the genitive, as it is governed by the numeral.

One solution would be to adopt a different dependency annotation scheme, e.g. one where numerals are heads of their governed nouns. However this would require training a parser on data annotated in such a scheme[26]. Moreover, the most popular syntactically oriented dependency annotation scheme — SUD, also attaches numerals as children of the nouns they govern. The native (non-UD) annotation scheme of PDB attaches governed nouns as children of numerals, but there is no spaCy model available at the moment, trained in this scheme.

Another solution, and this is the one that has been chosen in the final version, is to force the accommodating versions of the numerals while inflect-

[26]This is even more complicated as *pl_nask* is a transformer based model. The transformer layers themselves are modified during training, which entails, that all the annotating components (tagger, NER, dependency parser) have to be trained on the same data, at the same time. *pl_nask* was trained on NKJP in the UD format, which contains all three annotation layers. There are no such datasets for other dependency annotation schemes.

ing. It is favourable, because it does not modify the algorithm itself, instead it just modifies the morphology formalization, by adding a *deprel* → *forced attribute values* mapping. For the *nummod:gov*, a mapping *nummod:gov* → *congr* would be introduced. This leads to inelegant, but still somewhat common ways of connecting numerals with collective nouns. For instance, when inflecting the following into instrumental case:

Dziesięcioro drzwi i dwoje dzieci

ideally one would expect:

Dziesięciorgiem drzwi i dwojgiem dzieci

instead the system will produce:

Dziesięcioma drzwiami i dwoma dziećmi

Both the noun, and the numeral then share the case attribute (therefore case should be manually added to accommodated attributes for deprels representing these constructions), and so inflection can proceed as in other cases.

4 Evaluation

Evaluating NLP systems is usually done in a quantitative fashion, i.e. a big dataset of test cases is collected, and then the output of the system is compared to the predefined, expected (*gold*) outputs, and aggregated using established evaluation metrics. This procedure assumes that all the examples are equivalent, i.e. that they should contribute the same, towards the final score. Nevertheless one might expect that more intricate grammatical constructions would be rarer, and so handling them would be underemphasized by the evaluation procedure. For this reason, two methods of evaluation will be considered, one involving hand-crafted examples which are meant to reflect the scope of interest, and another which uses a much bigger test set. The precise results will be discussed below, but the four general criteria of evaluation listed in subsection 1.1, can now be assessed:

1. **Semantical correctness**: the system does not introduce, or remove any words, and so any semantic deviation must stem from failures to accurately capture the syntactic structure by the system.

2. **Syntactical correctness**: the system attempts to capture the syntactic structure of inflected phrases, albeit it is not faultless, the precise metrics will be listed below.

3. **Flexibility**: there is no explicit domain-dependency. The main gain in the domain of flexibility, is the extension of inflection into complex phrases, although the range of constructions which the system is capable of inflecting, is narrowed down. For words outside of the vocabulary, a neural component can be used instead.

4. **Control**: the system is predicatable and stable, mainly to the degree that other components are able to perform. The most important failures can come from lemmatization, and parsing errors.

4.1 Unit tests

The qualitative tests were done in the paradigm of unit tests, i.e. a set of test cases which are meant to reflect on some more challenging problems for an application, and ensure its proper functioning. These tests reflect the inflection algorithm itself, as all the cases where errors would be produced due to lemmatization, tagging, or parsing errors, were modified until correct analyses were given. For example the input phrase:

Polskie Górnictwo Naftowe i Gazownictwo S.A.

Had to be modified to:

Polskie Górnictwo Naftowe i Gazownictwo

Because '*S.*' was incorrectly lemmatized as an abbreviation for '*sekunda*'. In listing 4 all the test cases are listed. The dictionary-based single word inflection algorithm was used. The system passes all the tests.

4.2 Quantitative evaluation

In the quantitative evaluation, around $49k$ examples of phrases in different inflections were extracted from the lexicon **SEJF**[3]. Nominal phrases consisting of a noun and an adjective are the most common, but other, and more complex variable constructions are also present. Each inflected form of the phrase is paired with a tag, and a base form in the lexicon, and so they can be

```
("Uniwersytet Papieski im. Jana Pawła II", "gen", "Uniwersytetu Papieskiego im. Jana Pawła II")
("Maciej Kowalski", "dat:pl", "Maciejom Kowalskim")
("Anna Komorowska-Zielińska", "gen", "Anny Komorowskiej-Zielińskiej")
("Kalwaria Zebrzydowska", "dat", "Kalwarii Zebrzydowskiej")
("Królowa Elżbieta II", "inst", "Królową Elżbietą II")
("Królowa Elżbieta druga", "inst", "Królową Elżbietą drugą")
("PZPR", "dat", "PZPR-owi")
("PZPR", "nom", "PZPR")
("Góry Błękitne", "gen", "Gór Błękitnych")
("siódmy stycznia", "inst", "siódmym stycznia")
("godzina piętnasta", "pl:dat", "godzinom piętnastym")
("Polskie Górnictwo Gazowe i Naftownictwo", "inst", "Polskim Górnictwem Gazowym i Naftownictwem")
("Wydział Dziennikarstwa i Nauk Politycznych UW", "dat", "Wydziałowi Dziennikarstwa i Nauk Politycznych UW")
("Wojewódzki Ośrodek Ruchu Drogowego", "pl:gen", "Wojewódzkich Ośrodków Ruchu Drogowego")
("swoje własne dzieci", "gen:sg", "swego własnego dziecka")
("biała flaga z papieru", "gen:pl", "białych flag z papieru")
("biała jak sól, flaga z szorstkiego płótna", "pl:dat", "białym jak sól, flagom z szorstkiego płótna")
("połowa kilograma", "gen", "połowy kilograma")
("dwoje dzieci", "inst", "dwoma dziećmi")
("troje oczu", "inst", "trzema oczami")
("czworo młodych uszu", "loc", "czworgu młodych uszach")
("dziesięcioro skrzypiec", "loc", "dziesięciu skrzypcach")
("3 świnie", "loc", "3 świniach")
("cztery ściany", "dat", "czterem ścianom")
("trzy i pół tysiąca", "gen", "trzech i pół tysiąca")
("drzewa i dwoje małych dzieci", "dat", "drzewom i dwojgu małym dzieciom")
("trzy małe świnki", "dat", "trzem małym świnkom")
("sześć kwadratów i trójkątów", "loc", "sześciu kwadratach i trójkątach")
("sześć małych i złotych łańcuszków", "dat", "sześciu małym i złotym łańcuszkom")
("sto psów", "dat", "stu psom")
("milion złotych", "inst", "milionem złotych")
("jeden głupi wyraz", "loc", "jednym głupim wyrazie")
("pięciu łysych mężczyzn", "inst", "pięcioma łysymi mężczyznami")
("wysokiemu i czarno-zielonemu", "gen:pl", "wysokich i czarno-zielonych")
("biały jak kreda", "com", "bielszy jak kreda")
("biały i gładki", "sup:f", "najbielsza i najgładsza")
("pieprz, sól i mąka", "gen", "pieprzu, soli i mąki")
("kraina mlekiem i miodem płynąca", "pl:dat", "krainom mlekiem i miodem płynącym")
("czarny chleb i czarna kawa", "inst", "czarnym chlebem i czarną kawą")
("patrzeć", "ger", "patrzenie")
("zabijać", "ger:dat", "zabijaniu")
("zabijacie", "impt", "zabijajcie")
("zabijać", "impt:pl:pri", "zabijajmy")
("zabijać", "ppas:neg", "niezabijany")
("profesorowie", "depr", "profesory")
("biały", "adja", "biało")
("pies", "gen", "psa")
("człowiek", "pl", "ludzie")
("sitko", "inst", "sitkiem")
("widły", "pl:inst", "widłami")
("czeluść", "dat", "czeluści")
("świece", "dat:sg", "świecy")
("psom", "sg", "psu")
("piękny", "gen", "pięknego")
("zły", "pl", "źli")
("zwykły", "f", "zwykła")
("zwykłego", "f", "zwykłej")
("jasny", "f:pl:gen", "jasnych")
("ciemny", "f:pl:acc", "ciemne")
("białych", "sg:m3:acc", "biały")
("biała", "pl:m1", "biali")
("biała", "pl:m3", "białe")
("biała", "sup", "najbielsza")
("biegnący", "pl:gen", "biegnących")
("stojący", "f:neg:pl:dat", "niestojącym")
("stojący", "f:neg:pl:dat", "niestojącym")
("widzenie", "pl", "widzenia")
("widzenie", "neg", "niewidzenie")
("oddychanie", "dat", "oddychaniu")
("widzę", "pl", "widzimy")
("widzę", "ter", "widzi")
("widzę", "sec:pl", "widzicie")
("leczy", "pl", "leczą")
("leczy", "pl:sec", "leczycie")
("leczył", "f", "leczyła")
```

Listing 4: Unit test cases, the arguments in each tuple are as follows: the input phrase, the desired morphological profile, the expected output.

Base	Target pattern	Output	Gold	Comment
'*płacz i zgrzytanie zębów*'	LOC:M3	'*płaczu i zgrzytającym zębów*'	'*płaczu i zgrzytaniu zębów*'	Unnecessary agreement in the ruleset
'*warte lepszej sprawy*'	SG:INST:M1	'*wartym lepszym sprawy*'	'*wartym lepszej sprawy*'	Parser error
'*dziurawy worek*'	SG:DAT:M3	'*dziurawemu worek*'	'*dziurawemu workowi*'	Parser error
'*czapka uszanka*'	PL:ACC:F	'*czapki uszanka*'	'*czapki uszanki*'	No agreement in the ruleset
'*fitness club*'	PL:GEN:M3	'*fitness club*'	'*fitness clubów*'	No agreement in the ruleset
'*młode wilczyce*'	PL:GEN:F	'*młode wilczyc*'	'*młodych wilczyc*'	Parser/Tagger error
'*bilans czterolatka*'	PL:NOM:M3	'*bilanse czterolatka*'	'*bilansy czterolatka*'	Stylistic difference
'*ostry nabój*'	PL:GEN:M3	'*ostrych naboi*'	'*ostrych nabojów*'	Stylistic difference
'*happy end*'	SG:INST:M3	'*happy end*'	'*happy endem*'	Parser/Tagger error
'*zasadnicza rozmowa*'	SG:DAT:F	'*zasadniczej rozmowie*'	'*rozmowom zasadniczym*'	Annotation error
'*cioteczna siostra*'	SG:LOC:F	'*ciotecznej siostrze*'	'*siostrach ciotecznych*'	Annotation error

Table 5: Analysis of a random sample of errors on the SEJF lexicon.

used to evaluate the system. The phrases were inflected using the dictionary method, and no attempt was made to isolate the errors of the other components of the system. Therefore, the evaluation is reflective of the system as a whole, including the dependency parser, lemmatizer and tagger, as opposed to the inflection method in itself. The final accuracy equals **88.21%**, i.e. in 88.21% cases the output is (modulo permutations) orthographically identical to the inflected form listed by the dictionary. In Table 5 some examples of errors are listed, annotated with the discovered causes of errors.

5 Applications

The main application of the system: template-based NLG for Polish will now be discussed. In subsequent sections some additional use cases and extensions will be sketched out.

5.1 Parameterized Templates

The code in listing 6 combinatorically produces the messages listed in

```python
import random
import re
import spacy
from collections import Counter

nlp = spacy.load("pl_nask")
morf = nlp.get_pipe("morfeusz")
gap_finder = re.compile(r"#([^#:]+)(:[^#]+)?#")
gender_finder = re.compile(r"(?<=:)(m1|m2|m3|f|n)(?=:|\b)")
NOUNS = ["sweter", "bluza", "koszulka", "kimono"]
ADJS = ["czarny", "ekologiczny", "wykonany z wełny"]
PREPS = ["od znanego projektanta", "w stylu sportowym", "na co dzień"]
NUMS = ["trzy", "dwa"]
gap_to_examples = {"NOUN": NOUNS, "ADJ": ADJS, "PREP": PREPS, "NUM": NUMS}

def fill_template(input_template):
    matches = list(gap_finder.finditer(input_template))
    gap_types = Counter([m.groups()[0] for m in matches])
    fillings = {gap_type: random.sample(gap_to_examples[gap_type], k) for
    ↪  gap_type, k in gap_types.items()}
    filled_template = input_template
    for m in matches[::-1]:
        start, end = m.span()
        gap_type, inflection_pattern = m.groups()
        filling = fillings[gap_type].pop()
        if inflection_pattern:
            filling = morf.flex(nlp(filling),
            ↪  inflection_pattern.strip(":"))
        filled_template = filled_template[:start] + filling +
        ↪  filled_template[end:]
    return filled_template

for i in range(20):
    base = fill_template("#NOUN#")
    gender = gender_finder.search(nlp(base)[0].tag_).group(0)
    intermediate_template = fill_template(f"#ADJ:{gender}# {base} #PREP#")
    inflected_template = morf.flex(nlp(intermediate_template), "pl")
    intermediate_template = fill_template(f"#NUM# {inflected_template}")
    inflected_argument = morf.flex(nlp(intermediate_template), "loc")
    output = f"Myślę o {inflected_argument}"
    print(output)
```

Listing 5: Generating text by using the inflection component.

listing 5. It uses the `pl_nask` spaCy model, in which the inflection capabilities are implemented as part of the `morfeusz` pipeline component. The input text has to be first processed by the model, and then passed to the `flex` method. Some messages might be nonsensical, e.g. '*Myślę o dwu wykonanych z wełny kimonach w stylu sportowym*', but this is a matter of semantics, not of inflection. Here the feature combinations are purely random, but they could be constrained, e.g. to feature combinations actually occuring in the context (for example, a product database), in order to filter out nonsensical output.

The templates themselves are generated from the ground up dynamically, but they could be predefined with parameterized slots. The rough syntactical roles are given in capital letters, whereas any morphological constraints are given after ':'. If a template is more fleshed out beforehand, the gaps will usually need to be parameterized with these morphological constraints more densely.

Znaleźliśmy dla Ciebie cztery #NOUN:PL:ACC#*!*

```
"Myślę o dwu czarnych bluzach w stylu sportowym"
"Myślę o dwu czarnych kimonach na co dzień"
"Myślę o dwu czarnych kimonach w stylu sportowym"
"Myślę o dwu czarnych koszulkach w stylu sportowym"
"Myślę o dwu czarnych swetrach od znanego projektanta"
"Myślę o dwu ekologicznych koszulkach od znanego projektanta"
"Myślę o dwu ekologicznych swetrach w stylu sportowym"
"Myślę o dwu wykonanych z wełny bluzach w stylu sportowym"
"Myślę o dwu wykonanych z wełny swetrach na co dzień"
"Myślę o trzech czarnych kimonach na co dzień"
"Myślę o trzech czarnych kimonach od znanego projektanta"
"Myślę o trzech czarnych koszulce w stylu sportowym"
"Myślę o trzech czarnych swetrach na co dzień"
"Myślę o trzech ekologicznych bluzach na co dzień"
"Myślę o trzech ekologicznych koszulkach na co dzień"
"Myślę o trzech ekologicznych swetrach od znanego projektanta"
"Myślę o trzech ekologicznych swetrach w stylu sportowym"
"Myślę o trzech wykonanych z wełny kimonach w stylu sportowym"
"Myślę o trzech wykonanych z wełny koszulkach na co dzień"
# sorted, and uniq'ed
```

Listing 6: The generated messages.

5.1.1 Generating templates

The parameters in gaps specifying target morphology would normally have to be input by hand. But it is possible to partially automate this step by delexicalizing regular sentences, and marking the newly created gaps with features of the extracted words. For instance

Na co dzień, jeżdżę rowerem.

can be delexicalized with respect tot the noun '*rowerem*', and then parameterized by its morpological tag:

Na co dzień, jeżdżę #SG:INST:M3#.

which can be subsequently relexicalized. The entire procedure is described in listing 7. It should be observed however, that the tag includes gender, which in this case is not due to the context itself, but rather because of the initial filling of the gap. If a noun of a different gender (e.g. '*dorożka*') were to be put in it, no form satisfying M3 would be found, and so it would not be inflected. A mature solution would have to filter these morphological features, in order to parametrize the gap only with those, which are imposed by the context.

5.2 Extensions

In this section, some applications of the aforementioned tools and methods outside the main scope of interest will be described.

5.2.1 Derivation and inflection

It should be noted, that the phenomenon of inflection is closely related to *derivation*, the boundary between the two being rather blurry. Derivation is a heterogeneous set of mechanisms, which includes, among others:

- generation of diminutives, e.g. '*pies*' → '*piesek*'
- generation of names of professions or tools out or verbs '*piła*' → '*pilarz*'
- generation of nouns signifying possibility of some action '*palić*' → '*palny*'
- generation of new verbs by prepending them with prepositions '*prowadzić*' → '*przeprowadzić*'

```
def fill_single_template(template, filling):
    gap = gap_finder.search(template)
    target_feats = gap.groups()[1].strip(":")
    start, end = gap.span()
    doc = nlp(filling)
    inflected_filling = morf.flex(doc, target_feats)
    filled_template = template[:start] + inflected_filling +
    ↪  template[end:]
    return filled_template

doc = nlp("Na co dzień jeżdżę samochodem.")
gap_index = -2 # the gap token can be singled out using different means,
↪  e.g. its deprel
tag = doc[gap_index].tag_
pos, feats = tag.split(":", 1)
template = doc[:gap_index].text + f" #{feats}#" + doc[gap_index + 1:].text
fillings = ["motor", "samochód po dziadku", "powóz konnnym", "biały
↪  sedan", "miejski autobus lub tramwaj"]
for filling in fillings:
    print(fill_single_template(template, filling))

"Na co dzień jeżdżę motorem."
"Na co dzień jeżdżę samochodem po dziadku."
"Na co dzień jeżdżę powozem konnnym."
"Na co dzień jeżdżę białym sedanem."
"Na co dzień jeżdżę miejskim autobusem lub tramwajem."
```

Listing 7: Delexicalization of the input sentence, and filling the gap with different nominal phrases.

et cetera. Derivation, as opposed to inflection, generally includes a change in meaning, and is less regular and systematic[27]. These are also the reasons, for which including these in a template-based NLG system is not a very pressing concern, as they make delexicalization in the form of templates less viable. It would certainly be interesting to see a computer system capable of creative word formation, however this would require working within an entirely different framework, and a different set of tools. For this reason, derivation as such has been excluded from the scope of interest specified in subsection 2.3.

In SGJP lexemes include some forms, which belong to a different grammatical class, but are systematically derivable. For example gerunds and adjectival participles, are both grouped in the same lexeme, despite being recognized as independent grammatical classes. This grouping means, that gerunds and participles will be lemmatized to infinitive forms of the verbs they were derived from. For example "tańczenie" and "tańczący" will both be lemmatized to "tańczyć". This makes it important to distinguish a separate category of *flexemes*. A flexeme groups all the forms, which share the set of morphological attributes which they exemplify. Therefore "tańczący" and "nietańcząca" will belong to the same flexeme (as they both exemplify number, case, gender, aspect, and negation), but "tańczenie" will belong to a different one (because it has a fixed gender) and so will "tańczysz" (e.g. because it does not exemplify grammatical case).

In some applications it would be beneficial to be able to computationally derive forms which cross the flexeme boundary. It requires adding POS tags to the array of features which the algorithm can handle. Nevertheless an important provision has to be made. Crossing the flexeme boundary might introduce a new grammatical attribute to be selected for. E.g. while converting a verb '*nosić*' into a gerund, the categories of case, and negation are introduced, for which the input form has no value. Therefore forms such as '*nienoszenie*', '*noszeniu*' and '*noszenie*' are all equivalent. The original dictionary-based algorithm, has no way of preferring one of these, and it would be intuitive, that the affirmative form in the nomative were returned as the default one, unless specified otherwise by the user. This requires, that during inflection tags are enriched with default values for absent attributes,

[27]**Słowosieć** (`http://plwordnet.pwr.wroc.pl/wordnet/`) — the Polish version of **WordNet**, is an important step in the direction of systematizating derivation patterns. This year (2022) derivation in Polish (among others) was also one of the tasks of the yearly **SIGMORPHON** contest (`https://github.com/sigmorphon/2022InflectionST`).

attribute	def. value
case	nom
gender	m1
negation	aff
number	sg
degree	pos
aspect	ind
accommodability	congr

Table 6: Default values for attributes.

which is handled by the `enrich` keyword argument to the `tag_to_feats`. The default values are specified in Table 6. Some examples of single word inflections which cross the boundary of flexemes, are listed in listing 4

5.2.2 Lemmatization

To a certain degree inflection can be thought of as an inverse of lemmatization. The latter task can also be extended to phrases, which can be said to possess some privileged, base form. This is an important subject outside of NLG, for example in named entity normalization and resolution. Lemmatization of Polish phrases (mainly in the context of proper names) has been previously discussed by [8] and was one of the tasks proposed for the 2019 edition of the PolEval competition [9]. A trivial solution for phrase lemmatization can be based on concatenating lemmata of individual tokens. Unfortunately this rarely produces satisfying lemmata, as all the information about internal structure of the phrase is lost. For example:

Polskiej Rady Żywienia

will, after concatenation of lowercased lemmas produce:

polski rada żywić

instead of:

Polska Rada Żywienia

The problem is nearly identical to the one, which appeared while inflecting phrases. The root of the phrase could be lemmatized using standard means, but other words are bound by grammatical relations, which force taking up

particular forms. Ensuring that these constraints are not violated requires inflection of the remaining words in accordance with the changes resulting from lemmatization of the root. This entails, that a slight alteration of the phrase inflection algorithm can give capabilities of lemmatizing phrases.

```
def lemmatize_subtree(head):
    ind_to_lemmatized = {}
    children_to_lemmatize = list(head.children)
    target_feats = self.get_lemma_alterations(head)# recording
    ↪  all the morphological features that need to be changed
    ↪  to convert the input form into the lemma
    ind_to_lemmatized[head.i] = lemmatized_head
    lemmatized_head = self.flex_token(head, target_feats)
    for child in children_to_lemmatize:
        accommodable_feats = filter_accommodable_feats(child,
        ↪  target_feats)
        lemmatized_subtree = flex_subtree(child,
        ↪  accommodable_feats)
        ind_to_lemmatized.update(lemmatized_subtree)
    return ind_to_lemmatized
```

Listing 8: Function for lemmatizing phrases.

Initially, the root of the phrase is inflected to its lemma, and all the morphological alterations are registered. This is done via a mapping from POS tags, to lists of full morphological tags which a lemma might exhibit[28]. For instance: anterior participles with POS tag PANT have lemmas with either PANT:IMPERF or PANT:PERF, whereas lemmas for nouns exhibit a wider array of features, with different genders, numbers (in the case of *plurale tantum*) *etc.* Among these, the one with smallest size of symmetric difference with respect to the current tag is selected. The root is then inflected into this target tag, and all the morphological alterations with respect to the original tag, are propagated along dependency arcs, just as in the case of phrase inflection.

[28]It should be observed that this particular method, as opposed to traditional lemmatization, does not cross flexeme boundaries, e.g. adjectival participles will not be converted to infinitives. This is important, as such conversions might make the base form unrecognizable.

This method has been evaluated quantitatively on the same resource — SEJF. This time, different inflected forms were put in as input, with the base forms being the target forms. The algorithm achieves **78.93%** accuracy, which is a significant improvement, over the baseline (concatenating lemmas) achieving **36.06%**. It should be noted however, that this algorithm is not well suited to lemmatizing phrases, which exhibit plural number in their base form, as is the case in most phrases involving numerals. This is because lemmatizing the root usually involves converting it into the singular form.

5.2.3 Extending into other languages

To an important degree the inflection algorithm described above is language agnostic, and so it is natural to attempt to extend it into other languages. To test the feasibility of working in a different language, a similar solution was prepared for Russian. Russian is closely related to Polish, and also exhibits rich morphology (being a fusional language, like Polish), but employs a different alphabet. The main difficulty in implementation lied in finding compatible resources. The Russian spaCy model is trained on a non-official silver-standard corpus[29]. The morphological layer in this corpus is annotated in UFEATS tagset, and so the agreement statistics and rules were formulated in this tagset aswell, by extracting them based on a subset of documents in the corpus. It was not possible to find a dictionary for Russian in this tagset, but because of the size of the corpus, it was possible to generate such a dictionary based on the forms in the data. Unfortunately the corpus is not annotated with lemmata, so this information had to be added *via* external resources: the **pymorphy2** package[30].

Regrettably, a manual analysis of the general performance of the system shows that it is significantly less reliable than that of the Polish application. For this reason a quantitative analysis of the entire system, which would not be able to isolate the inflection component itself, was not undertaken. Nevertheless, in listing 9 some examples of successful inflections are shown, to demonstrate that transfer is generally possible. A more mature solution would require obtaining a better set of resources, perhaps specified in a native tagset for representing Russian morphology.

[29] `https://github.com/natasha/nerus`

[30] `https://github.com/kmike/pymorphy2`

```
("абсолютный падеж", "Case=Ins|Number=Sing")
#абсолютным падежом
("активный словарный запас", "Case=Dat|Number=Sing")
#активному словарному запасу
("антенна радиолокационного дальномера", "Case=Loc|Number=Sing")
#антенне радиолокационного дальномера
("брак по договорённости", "Case=Gen|Number=Plur")
#браков по договорённости
("буря в стакане воды", "Case=Ins|Number=Plur")
#бурями в стакане воды
("физика элементарных частиц", "Case=Loc|Number=Sing")
#физике элементарных частиц
("гражданская война в США", "Case=Dat|Number=Sing")
#гражданской войне в США
("книга за семью печатями", "Case=Ins|Number=Plur")
#книгами за семью печатями
("компенсационное натяжное колесо", "Case=Gen|Number=Sing")
#компенсационного натяжного колеса
("многоразовый транспортный космический корабль",
↪  "Case=Dat|Number=Plur")
#многоразовым транспортным космическим кораблям
```

Listing 9: Some examples of the system inflecting bases (the first string) into desired UFEATS patterns (second string). System outputs are in the commented lines below. The examples were taken from the Russian Wiktionary

6 Concluding remarks

This thesis aimed to discuss an implementation of the system for generating inflections for the purposes of Natural Language Generation. Two methods for inflecting individual words were discussed, with attempts to evaluate them. It has been shown, that the neural model has an area of an application in inflecting words outside of the dictionary, but is generally inferior to the dictionary method. The methods for inflecting individual words were then integrated into an algorithm for inflecting phrases. It has been shown, that to a useful degree, agreement relations (which are an important factor in inflecting phrases) can be approximated with dependency analyses of the phrases. This method of representing their structure, allows to inflect a broad range

of linguistic constructions without violating syntactic constraints. In the end, the system exhibits a rather classical structure, using rules, recurrent data structures, and dictionaries.

The main application of the system has been discussed, with examples in code, as to how to use it for NLG. Some extensions for basic derivation, lemmatization of phrases, and support for languages other than Polish, were sketched out together with the necessary provisions in the code of the system.

The methods presented here, can also be thought of as parts of a larger suite of tools for NLG, which could contain other algorithms, such as tools for constructing numeral phrases out of numbers represented digitally, or a more robust system for handling verbal paradigms, and sentence transformations in general. These areas however, are much more difficult to handle in a language-agnostic manner, as the standard approach to segmentation and morphological treatment of Polish verbs, and numerals is quite intricate.

References

[1] Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels, October 2018. Association for Computational Linguistics.

[2] Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver, August 2017. Association for Computational Linguistics.

[3] Czerepowicka, Monika and Savary, Agata. SEJF - A Grammatical Lexicon of Polish Multiword Expressions. In Vetulani, Zygmunt and Mariani, Joseph and Kubis, Marek, editor, *Human Language Technology. Chal-*

lenges for Computer Science and Linguistics, pages 59–73, Cham, 2018. Springer International Publishing.

[4] Ondrej Dusek and Filip Jurcícek. Neural generation for czech: Data and baselines. In Kees van Deemter, Chenghua Lin, and Hiroya Takamura, editors, *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 563–574. Association for Computational Linguistics, 2019.

[5] Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. Morphological inflection generation using character sequence to sequence learning. *CoRR*, abs/1512.06110, 2015.

[6] Witold Kieraś and Marcin Woliński. Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego. *Język Polski*, XCVII(1):75–83, 2017.

[7] Witold Kieraś, Marcin Woliński, and Bartłomiej Nitoń. Nowe wielowarstwowe znakowanie lingwistyczne zrównoważonego narodowego korpusu języka polskiego. *Język Polski*, (2):59–70, wrz. 2021.

[8] Michał Marcińczuk. Lemmatization of multi-word common noun phrases and named entities in Polish. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 483–491, Varna, Bulgaria, September 2017. INCOMA Ltd.

[9] Maciej Ogrodniczuk and Łukasz Kobyliński, editors. *Proceedings of the PolEval 2019 Workshop*, Warsaw, Poland, 2019. Institute of Computer Science, Polish Academy of Sciences.

[10] Tiago Pimentel, Maria Ryskina, Sabrina J. Mielke, Shijie Wu, Eleanor Chodroff, Brian Leonard, Garrett Nicolai, Yustinus Ghanggo Ate, Salam Khalifa, Nizar Habash, Charbel El-Khaissi, Omer Goldman, Michael Gasser, William Lane, Matt Coler, Arturo Oncevay, Jaime Rafael Montoya Samame, Gema Celeste Silva Villegas, Adam Ek, Jean-Philippe Bernardy, Andrey Shcherbakov, Aziyana Bayyr-ool, Karina Sheifer, Sofya Ganieva, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Andrew Krizhanovsky, Natalia Krizhanovsky, Clara Vania, Sardana Ivanova,

Aelita Salchak, Christopher Straughn, Zoey Liu, Jonathan North Washington, Duygu Ataman, Witold Kieraś, Marcin Woliński, Totok Suhardijanto, Niklas Stoehr, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Richard J. Hatcher, Emily Prud'hommeaux, Ritesh Kumar, Mans Hulden, Botond Barta, Dorina Lakatos, Gábor Szolnok, Judit Ács, Mohit Raj, David Yarowsky, Ryan Cotterell, Ben Ambridge, and Ekaterina Vylomova. SIGMORPHON 2021 shared task on morphological reinflection: Generalization across languages. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–259, Online, August 2021. Association for Computational Linguistics.

[11] Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. *Narodowy Korpus Jezyka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw, 2012.

[12] Zygmunt Saloni and Marek Świdziński. *Składnia współczesnego języka polskiego*. Polskie Wydawnictwo Naukowe, Warszawa, 2012.

[13] Agata Savary, Joanna Rabiega-Wiśniewska, and Marcin Woliński. Inflection of Polish Multi-Word Proper Names with Morfeusz and Multiflex. In Małgorzata Marciniak and Agnieszka Mykowiecka, editors, *Aspects of Natural Language Processing, Lecture Notes in Computer Science 5070*, pages 111–141. Springer Verlag, 2009.

[14] Andrei Shcherbakov, Saliha Muradoglu, and Ekaterina Vylomova. Exploring looping effects in RNN-based architectures. In *Proceedings of the The 18th Annual Workshop of the Australasian Language Technology Association*, pages 115–120, Virtual Workshop, December 2020. Australasian Language Technology Association.

[15] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, jan 1966.

[16] Marcin Woliński. Morfeusz reloaded. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors,

Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, pages 1106–1111, Reykjavík, Iceland, 2014. European Language Resources Association (ELRA).

[17] Marcin Woliński. *Automatyczna analiza składnikowa języka polskiego*. Wydawnictwo Uniwersytetu Warszawskiego, Warszawa, 2019.

[18] Alina Wróblewska. Extended and enhanced Polish dependency bank in Universal Dependencies format. In Marie-Catherine de Marneffe, Teresa Lynn, and Sebastian Schuster, editors, *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 173–182. Association for Computational Linguistics, 2018.