Jens Kolb et al. (Eds.): Enterprise Modelling and Information Systems Architectures,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 17

Flexible Evaluation of Textual Labels in Conceptual Models

Arian Storch[1], Ralf Laue[2], Volker Gruhn[3]

Abstract:

This paper introduces a flexible and generic approach to define customised style rules for labels in conceptual models. A rule-based language is presented which can express style rules in a flexible and context-specific way. The formalised style rules are used to analyse and evaluate textual labels of model elements. To analyse a textual label, a combination of standardised natural language processing tools such as a part-of-speech tagger or a named entity recogniser are used. With the help of these techniques, custom-defined information entities can be extracted from the model.

Keywords: Conceptual Modelling, Natural Language Processing, Configuration, Customisation, Information Entity Extraction

1 Introduction

Conceptual models are typically used to document and communicate the architecture, environment and processes of an organisation [HWPZ03]. Once created, they can be analysed, discussed and gradually changed or improved to fit the constantly changing needs. Furthermore, they can be used as a source for designing and implementing software applications. In order to serve as a means for communication, the models have to be not only correct, but also easy to understand. Therefore, domain experts, companies and researchers have defined modelling guidelines with the aim to improve the quality and understandability of models [Si11].

[LSS94] elaborated three quality characteristics of conceptual models: *syntactic quality*, *semantic quality* and *pragmatic quality*. One aspect of *pragmatic quality* is *label quality*, and one measurable aspect of *label quality* is the adherence to naming conventions [SLG13]. Enforcing naming conventions can help to avoid misunderstandings. Mendling et al. [MRR10] give several examples of labels that can raise understanding problems when no naming convention is in use. For example, for "measure processing" it is unclear whether "to measure" or "to process" is the verb describing the action. Other authors emphasise the importance of text labels for the understanding and comparison of process models [MRR10, Be09b, NH15]. When models have to be compared (for purposes such as benchmarking, compliance analysis or model merging) it is necessary to know which nodes correspond to each other – even if a node is labeled "test software" in one model and

[1] it factum GmbH, Hainstr. 6, 04109 Leipzig, Germany, arian.storch@it-factum.de
[2] University of Applied Sciences of Zwickau, Department of Computer Science, Dr.-Friedrichs-Ring 2a, 08056 Zwickau, Germany, ralf.laue@fh-zwickau.de
[3] Paluno - The Ruhr Institute for Software Technology, University of Duisburg-Essen, Gerlingstr. 16, 45127 Essen, Germany, volker.gruhn@paluno.uni-due.de

"software testing" in another one. The same is true for approaches such as [HD15] where process errors are found based on patterns which depend on the actions expressed by the element labels. For such purposes, it is necessary to extract information entities from a label. In the given example, one wants to know that "to test" describes the activity and "software" is the object.

Recommendations for naming conventions are typically grounded on empirical studies and related to a modelling language, but not to a business or organisation. However, in certain domains, there can be special demands for label styles which are not covered by existing naming conventions. For example, Combi et al. [Co12] suggest the modelling of medical processes with activity labels that can contain time information such as "Patient Evaluation [5,20] min". Obviously, such a label would not fulfill any of the commonly suggested style rules. Therefore, tools which can check only whether an activity label follows some "standard" style, would recognise this label as *irregular*. Therefore, what is needed is an approach that allows organisations to define own naming conventions.

In this paper, we introduce a flexible and generic approach that allows users to define style rules for model element labels according to their specific needs. These style definitions are expressed using a proprietary part-of-speech (POS) pattern description language. This language also supports the extraction of specific semantic word groups of a label text (such as a business object from an activity label). Based on these formalised label style rules, we build up an algorithm to analyse and evaluate labels of model elements. We utilise a combination of natural language processing (NLP) tools such as a POS tagger or named entity recognition (NER) to determine the POS of words and validate them against the formalised label style rule. Making use of the fact that there are mature NLP tools for the English language, we implemented an algorithm for analysing English model element labels.

This paper is structured as follows. First, we describe the idea of part-of-speech tagging, the NLP technique on which our approach is grounded. Next, we analyse the related work about label analysis and discuss the research gap that resulted in the approach presented in this paper. In section 3 we explain our approach and discuss its advantages and shortcomings. Finally, we summarise the work in section 4 and motivate further research.

2 Background

2.1 Part-of-Speech Tagging

NLP refers to a theoretically motivated range of computational techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing [Li01]. In the context of conceptual models, one field of application of NLP is to recognise the part-of-speech (POS) of words and phrases in a label of a model element. Based on this information, the word sequence can be decomposed into information entities, e.g. the activity (verb) or the affected subject (object).

The process of determining the POS of words and phrases is called POS tagging. POS taggers typically combine lexical databases with statistical algorithms to determine the kind of a word or phrase within a sentence [Me02]. One of the most powerful and popular POS taggers is the Stanford Tagger[4] which is part of a toolset developed by the Stanford Natural Language Processing Group[5]. Its tagging methods are based on *statistical parsing*, where the frequency of grammar rules is exploited for finding the most probable POS of a word. These frequencies were obtained from a manually preprocessed (hand-parsed) collection of texts, called *text corpus*.

However, even though NLP tools provide reliable results when being applied to natural language texts [To03] they have to deal with difficult issues when processing labels in conceptual models. In most cases, such labels consist of one up to three words and do not fulfill a complete sentence structure [LSM12]. This makes it difficult to find the POS of a word, given the fact that in English derivation (such as from a noun to a verb) can occur without any change of form (this phenomenon is called *conversion* or *zero derivation*) [Di08]. For instance, the English word "log" can be a verb or a noun. Additionally, the accuracy of NLP tools decreases if the label contains special characters, for instance "Read/save/print notification [A5 page size]".

2.2 Related work

Combining techniques of NLP tools with conceptional models has been applied to many fields. On the one hand, there are several approaches to create conceptional models from natural language text or vice versa. Montes et al. [Mo08] use a POS tagger and parser to automatically generate a conceptual model from the textual descriptions of use case scenarios. Other authors are using parsers and WordNet[6] (a lexical database which provides information about semantic and lexical relations between words [Fe98]) to create BPMN Process Models, ER-Models or UML-Models from text [FMP11, GSD99, BC12]. Approaches to generate text from conceptual models are described in [Da92, MAA08, LRR96]. All these approaches have in common that they use a conceptual model either as source or as target of a transformation. On the other hand, there are algorithms for inspecting a single model. Some authors developed approaches to increase the understandability and consistency by reducing linguistic variations and enforcing naming conventions of model element labels. This is done by relying on WordNet or domain-specific terminology databases [vdVGvdR97, KHO11, Be09b]. Other authors are using labels for measuring the quality or similarity [EKO07, NH15] or for detecting semantic errors [GL11, HD15]. A detailed overview of existing approaches can be found in [Le13].

Reducing naming variations and enforcing style guidelines can help to avoid errors and misunderstandings. Therefore, modelling and label style guidelines have been developed. Algorithm and tools can be designed to determine and enforce the compliance to such style rules. In the case of business process models, Leopold et al. [LSM11] introduced

[4] http://nlp.stanford.edu/software/tagger.shtml
[5] http://www-nlp.stanford.edu/
[6] http://wordnet.princeton.edu/

an algorithm to recognise the style of labels for describing activities. Assuming that there are seven labeling styles used in business process models [LSM12], they designed an algorithm to recognise the label style by comparing words, their order within the word sequence and their POS to grammatical phrase structures that have been derived from the style rules. The POS determination is done with the help of WordNet and the Stanford Tagger. If a label can not be recognised clearly, it is examined whether this word can be found and assigned to its POS in other labels of the same model or in other models in a repository. However, Leopold et al. do not recommend the application of the Stanford Tagger (cf. [LSM12, LSM09]) because labels often do not meet the requirement of proper sentences. Anyhow, they observed a considerable increase of accuracy when using complements to extend the label text to a full sentence [LSM09]. It has to be noted that in the tools described [LSM12, LSM09, LSM11] the allowed style guidelines are hard-coded in the software. This does not allow easy customisation to specific needs or adaption to new modelling languages whose model elements express other concepts.

In [Le13], Leopold et al. describe impressive results for validating labels in business process models. They used linguistic patterns built from sequences of POS to operationalize style rules. A desirable style for activity labels of business process models (called *verb-object* style) was defined as *"Verb(Imperative) + Noun [+ Preposition + Noun]"* (square brackets denote optional elements). When analysing whether activities in the SAP reference model adhere to this style, they achieved an F-measure (the harmonic mean of precision and recall) of 96.7%. The algorithm presented in [Le13] relies on manually tagged corpora and can therefore be used even if no other NLP tools for a language exist. It first determines the POS of each word and then checks whether this sequence of POS corresponds to the defined style. As an example, the label "Provide service" would be classified as correct (a verb followed by a noun), but "Project planning" won't. Additionally, Leopold et al. [Le13] exploit information about labels of other model elements and from other models in a model repository. On the downside, such information is not available when a new model is created and no model repository exists so far.

Becker et al. [Be09a] introduced an approach to define naming conventions based on a linguistic grammar which describes phrase structures. This approach is applicable for any modelling language. With the help of the grammar, particular style rules can be expressed. As an example, the expression $< verb, imperative >< noun, singular >$ can be used to define a label style for a process activity. The POS determination and expression matching is realised by utilising a domain-specific terminology database and word relations (for instance, synonyms) queried from WordNet. By restricting the set of available words by the terminology database, there is no need to use additional NLP tools. If a word and its POS can not be analysed automatically because of unknown words or missing relations, the modeller has to interact with the modelling system to resolve the problem [Be09b].

Flexible style rules have been used to define linguistic patterns for describing natural language requirements by de Almeida Ferreira and da Silva [dd13]. However, their approach for checking the style cannot directly be transferred to conceptual models, because de Almeida Ferreira and da Silva make use of a deterministic set of keywords frequently occurring in requirements (for instance, "x IS y" or "x HAS y").

Object	POS tags (see Tab. 2)
Cheque	NN
Salary Cheque	NN NN
Valid cheque	JJ NN
A complete list of overdue salary cheques	DT JJ NN IN JJ NN NNS

Tab. 1: Various Word Sequences Forming an Object

2.3 Motivation

The related work discussed in section 2.2 shows that there is quite some research in the field of analysing labels of conceptual models. However, we still observe a notable gap when adapting common style guidelines to concrete domains, for example in order to allow labels such as "Patient Evaluation [5,20] min" mentioned in Sect. 1.

By studying labels of different conceptual model collections we observed that even without such specific style guidelines, the pattern *"Verb(Imperative) + Noun [+ Preposition + Noun]"* used in [Le13] for describing an activity would be far too strict. As an example, the label "Choose non-available item" would be classified as being wrong because adjectives are not permitted. But in our opinion, this label should be regarded as having the *verb-object* style as well. Moreover, we noted that from a linguistic point of view an object can consist of a wide range of words with different POS. For instance, the label "print list of overdue cheques" contains the object "list of overdue cheques" which is built from multiple nouns, a preposition and an adjective. Some kinds of objects and their POS are shown in Tab. 1. The tags are explained in Tab. 2. Considering the variety of conceptual models and domain specific needs, we believe that there is a need of a flexible linguistic pattern expression language.

As already mentioned by other researches, one main problem of POS determination is the fact that words sharing the same form belong to different word classes. For instance, the word "test" can be a noun or a verb. To obtain a correct decision, more context information is needed. One type of context can be the surrounding words. Using a word sequence such as "test the software" allows classifying the word correctly. Another type of context is the type of the model element carrying the label. For instance, the label "Review" gets a different understanding when being used in an activity or a resource model element respectively.

Becker et al. [Be09a] avoid the problem of words sharing the same form by requiring that all words used in a label must be taken from a set of words defined by a domain-specific terminology database. We acknowledge the advantages of using such a terminology database. However, in many situations maintaining such a database will not be feasible.

In order to offer the possibility for checking labels styles in as many situations as possible, it was our aim to create an approach that...

- allows to define the style rules in a flexible manner,

- does not restrict the vocabulary by requiring that all words belong to a terminology database, and
- does not require the presence of a model repository.

3 Customised Pattern-Based Label Evaluation

In this section, we present our approach to increase the reliability and flexibility of a textual label analysis. First, we introduce a pattern description language that is used to define the expected grammatical phrase structure and POS of word sequences in a model element label. Then, we describe the three stages of our algorithm. Figure 2 illustrates these stages; the steps shown in this figure will be explained in the following subsections.

3.1 Pattern Expression Language

Tag	Part-of-Speech
CC	coordinating conjunction
DT	Determiner
EX	Existential there
IN	preposition
JJ	Adjective
NN	Noun
VB	Verb
VBN	Verb, past participle
VBZ	Verb, 3rd person singular present

Tab. 2: Short Extract of the Penn Treebank Tag Set

In our previous work [SLG14], we gave examples how patterns for label styles can be expressed using a combination of tags from the Penn Treebank Tag Set (PTTS) and the Extended Backus-Naur Form (EBNF, ISO standard 14977). The PTTS defines a standardised set of tags denoting the POS of a word that has been processed by a POS tagger [Sa90]. Table 2 shows a subset of the tags defined in the PTTS. With the help of this notation, we defined style rules for the goal-oriented modelling language i^*. In subsequent work, some elements of the style rule specification language have been redefined in order to improve the readability, modularity and flexibility.

Listing 1 and 2 show the style rule specifications for a task and a goal in the language i^*. For example, the rules for task labels define that the text must start with a verb in base form and may be followed by a conjunction and another verb in base form[7]. Then, there must be a word sequence optionally headed by a preposition that matches the object rule. Optionally, an additional conjunction followed by an object may complete the label text.

[7] One might argue that it is not a good idea to allow two verbs in a label [BK04], but our point is that such a rule should not be generally defined but decided by the organisation according to their specific needs.

For example, a task label could be: "**debit** (VB) **the credit card** (object, DT NN NN)" while a goal label according to Listing 2 could look like: "**transfer** (object, NN) **is completed** (VBN)".

```
task {
 nnseq          = (NN|NNS)+;
 attrNoun       = DT? JJ* VBN? nnseq;
 object         = attrNoun ((IN|TO) attrNoun)?;
 verbImp        = VB (RP|IN)?;
 task           = verbImp (CC verbImp)? (IN|TO)? object
                  (CC object)?;
  []            = "You have to $.";
}
```

List. 1: Task Style Rule

```
goal {
 nnseq          = (NN|NNS)+;
 attrNoun       = DT? JJ* VBN? nnseq;
 object         = attrNoun ((IN|TO) attrNoun)?;
 goal           = object (CC object)? ("is"|"are") VBN;
 []             = "The $.";
}
```

List. 2: Goal Style Rule

A rule definition starts by naming the process model element to which the rule shall be applied to. Then, within the curling brackets, sub-rules are defined. Though, different rules can share the same sub-rules without re-defining them, the declaration here has been made redundant to clarify the intention. Table 3 lists the meaning of the special characters.

+	At least one occurrence
*	Zero or more occurrences
?	Optional occurrence
(...)	Groups occurrences or expressions
"..."	Exact match
... | ...	Alternative match; At least one expression must match (within a group)
$	Origin label text

Tab. 3: Overview of the Special Characters to express Operators

As already mentioned, POS taggers can produce more reliable results when operating on full sentences. Therefore, our style rules define a prefix so that the label text can be completed to full sentences (in the expression after the "[]"symbol). The special character "$" defines the location to insert the original label text. For instance, the activity label "Complete first test" will be complemented to "You have to complete first test.". A text matches the style rule if and only if it becomes a full sentence when completed using the given complement.

When analysing real-world model element labels, we realised that the use of some words or phrases can lead to problems. For example, in an activity label “perform investigation”, it would be wrong to conclude that “perform” is the verb and “investigation” is the business object. Instead, verbs such as “perform”, “execute” etc. should be avoided, and the text should be “investigate [object]” instead. Similarly, in a model in the language i^* , we do not want to have labels such as “achieve [something]” (which would rather be a goal than an activity); or in VDML models (describing the process of value delivering), an activity label should not be “provide [something]” (which would rather be a value proposition element). For this reason, our rule definitions allow to exclude certain words or word groups, using constructs such as

```
verb = VB;
verb != "perform" | "execute";
```

In this way, our language can use both a whitelist (listing all words or word groups that are allowed at a certain position) and a blacklist (listing all words that are not allowed, despite the fact that their use would be correct from a grammatical point of view).

There are further notation elements to express prefix and postfix of words, words surrounded by special characters, etc. Due to space limits, we won’t give a full overview here.

For applying the style rules, we transform them into regular expressions. This allows us to use standard library functions for checking whether the output of a POS tagger (and hence the label) conforms to the style rule. For example, the rules of Listing 3 are transformed into the regular expression *ˆ(/DT[ˆ/A-Z]*)?(/JJ[ˆ/A-Z]*)?(/NN[ˆ/A-Z]*)+$.*

```
nnseq          = NN+;
object         = DT? JJ? nnseq;
```

List. 3: Object style rule excerpt

All existing style rules will be aggregated and analysed. For each model element type, the result will be written in a label metadata repository. Figure 1 illustrates this process.

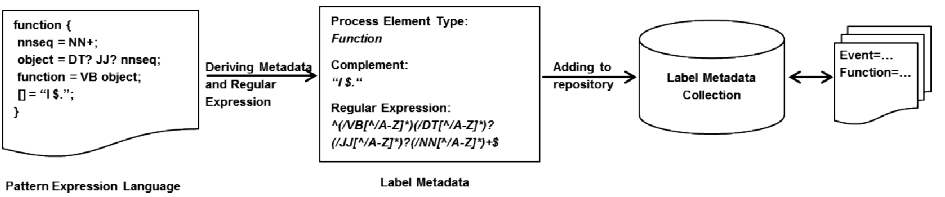

Fig. 1: Creating Label Metadata

3.2 Evaluation of Phrase Structure and POS of a Model Element Label Text

This section outlines our approach to evaluate the phrase structure and POS of model element labels. The designed algorithm consists of three stages as illustrated in Figure 2.

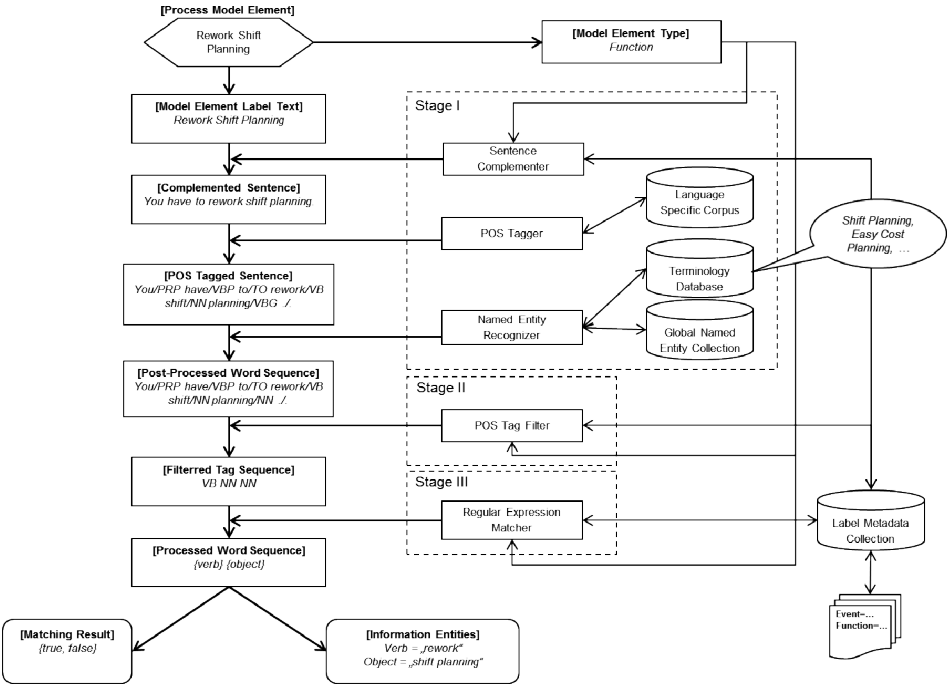

Fig. 2: Process Stage Model

Stage I: Sentence Completion / POS Determination / Named Entity-Recognition (optional) When analysing a label, we start by building a sentence using the sentence complement defined for the model element type. Then, the POS tagger is applied to determine the POS of each word of the text. Afterwards, a *named entity recogniser* (NER) can be used to improve the sequence of POS tags. It looks up words and word groups in a domain-specific terminology database which might be created by domain experts beforehand and re-tags it if necessary. Such a terminology database may also include domain-specific abbreviations. Additionally, the recogniser uses a collection of globally known named entities to regard non-domain-specific named entities (e.g., names of locations, organisations, titles of laws, etc.), too.

Stage II: POS Tag Filtering After each word has been annotated with its most likely POS, we have to apply a *tag filter*. It removes irrelevant POS tags for those words which have been added as a complement to the original label in stage I. At the end of stage II we have a tag sequence that represents the POS tags of the label.

Stage III: Regular Expression Matcher A *regular expression matcher* is finally used to compare the actual POS tag sequence with the expected one. Only if the sequence matches the regular expression, we conclude the label to be valid. In this case, we can additionally

extract the phrases associated with a matching sub-rule, for example we can conclude that in “approve all valid insurance claims” the object is denoted by the phrase “all valid insurance claims”.

3.3 Analysis Results

To validate our approach at an early development stage, we analysed two collections of labels taken from different model collections. The first is a collection of 100 task labels taken from an i^* model repository. The second is a collection of 100 EPC function labels taken from the SAP reference model. We analysed these collections twice. The first time, we used a derived definition of the *verb-object* style as given in [Le13]: *“Verb(Imperative) + Noun [+ Preposition + Noun]”*. Its formalisation in our rule language is shown in List. 4. Though Leopold et al. mentioned composite nouns (e.g., service order), it is not fully described how they recognise them. Therefore, we decided to use a pattern which conforms to the given formal definition. The second time, we used the definition according to List. 1. Following these definitions, we manually classified each label within these collections. We then compared the manual results with those of our algorithm. Tab. 4 gives an overview of the collections and the running time of our algorithm on a Lenovo X1 Carbon 2014 with a 1.50 GHz Intel Core i7-4550U processor, 8 GB RAM and a SSD device, running on Windows 7 and a JVM 1.8. The initialisation time of the Stanford POS Tagger and WordNet has not been measured.

```
activity {
 noun            = (NN|NNS);
 prep            = (IN|TO);
 activity        = VB noun (prep noun)?;
}
```

List. 4: Strict Verb-object Style according to Leopold et al.

As shown in Tab. 4, both collections have very different numbers of labels written in *verb-object* style. Due to the strict definition of the style rule following [Le13], many labels do not match. E.g., “Get relevant items” does not match because it contains an adjective. By contrast, the more tolerant style rule according to List. 1 allows amongst others the existence of adjectives before a noun. Therefore, the number of labels adhering to this style rule is much higher.

As part of our evaluation, we run our algorithm four times for each style rule definition. We used the Stanford Tagger as POS tagger. For correctly classifying named entities and business terms, we made use of the glossary in the terminology base sapterm.com which lists various common business terms. Additionally, we utilised WordNet to correct POS tags of words that may have been tagged wrongly by the Stanford Tagger (due to the used complement) but are actually unambiguous.

We started without a prefix or NER. Then, we successively added each feature, using “You have to $.” as prefix. The results of each evaluation are shown in Tab. 5 and 6.

Model collection	i^*	SAP
	100 i^* Tasks	100 EPC Functions
Average no. of words per label	2.98	3.23
Minimum no. of words per label	1	1
Maximum no. of words per label	7	9
No. of labels in verb-object style		
As defined in Listing 4	38	5
As defined in Listing 1	80	28
Performance results applying Listing 1		
Avg. running time per label (ms)	1.38	2.02
Max. running time per label (ms)	2.0	52.0

Tab. 4: Model Collections Details and Performance Results

Model Collection	i^*			SAP		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
No prefix, no NER	100.0 %	63.2 %	77.4 %	-[8]	0.0 %	-
No prefix, NER	100.0 %	71.1 %	83.1 %	-[8]	0.0 %	-
Prefix, no NER	100.0 %	86.8 %	93.0 %	33.3 %	100.0 %	50.0 %
Prefix, NER	100.0 %	97.4 %	98.7 %	83.3 %	100.0 %	90.9 %

Tab. 5: Evaluation Results of Verb-Object Style according to List. 4

Model Collection	i^*			SAP		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
No prefix, no NER	100.0 %	67.5 %	80.6 %	100.0 %	28.6 %	44.4 %
No prefix, NER	100.0 %	73.8 %	84.9 %	100.0 %	32.1 %	48.6 %
Prefix, no NER	100.0 %	90.0 %	94.7 %	65.8 %	89.3 %	75.8 %
Prefix, NER	100.0 %	98.8%	99.4 %	100.0 %	96.4 %	98.2 %

Tab. 6: Evaluation Results of Verb-Object Style according to List. 1

We assessed the accuracy of our algorithm using the metrics precision, recall and F-measure (harmonic mean of precision and recall). According to the results, the combination of using a prefix and NER gains best results. In this case, the lowest F-measure is 90.9 % when applying the strict *verb-obect* style rule to the SAP label collection. The best result is gained when applying the more tolerant *verb-object* style rule to the i^* label collection.

We identified two issues that may significantly reduce the accuracy of the analysis. First, if a word is misspelled, it is not possible to correctly determine its POS by the POS tagger because it can not be found in the corpus. For instance, if a word sequence is given as "check paymend" instead of the correct form "check payment", the resulting POS tag sequence is "*VB VBD*" instead of the correct one "*VB NN*". We can observe another aspect of misspelling when the change of just one character leads to another POS or even word

[8] Neither True nor False Positives have been measured

meaning. E.g., “write advice” gets the tag result “*VB NN*” while “write advise” is tagged as “*NNP VBP*”.

Another issue is the text corpus that has been used to build/train the POS tagger. POS tagging works such that a POS with a high occurrence rate within the text corpus is preferred. But this occurrence rate is mainly influenced by the manually annotated texts in the corpus. Therefore, different corpora may lead to different tagging results. In general, we assume that corpora built on texts out of business domains will lead to more reliable results.

4 Conclusion and Future Work

In this paper, we present an approach to define highly customisable style rules for model element labels. These rules are expressed using a pattern-based language. It allows domain experts or modellers to define style rules according to their specific needs. We believe that our pattern expression language is very flexible, well to read and to understand. Therefore, it may be helpful in the context of end-user-programming.

One main benefit of our approach is that it can be used without referring to a model repository or a lexical database. Furthermore, it is fully automated and needs no user input once the style rules have been defined. Complements added to the labels provide the possibility to build proper sentences that can be processed by the POS tagger. In addition, the accuracy can be improved by using NER and a terminology database.

By applying standard NLP tools to the label, the POS of each word can be determined, and it can be checked whether a label conforms to a style rule. With the help of custom definable sub-rules, information entities can be extracted from analysed labels. This offers the possibility to transform a model (e.g., into another conceptual model type or natural text) or to analyse the label parts in another way.

The accuracy of this approach is mainly influenced by the spelling of the textual labels. Therefore, we plan to add additional preprocessing stages to our algorithm in order to deal with misspelled words. In addition, we plan to add the possibility to transform a label from a given custom style into another style to support the maintenance of model repositories. This will be important for the purpose of merging models that have been created in different organisations.

References

[BC12] Bajwa, Imran Sarwar; Choudhary, M Abbas: From natural language software specifications to UML class models. In: Enterprise Information Systems, pp. 224–237. Springer, 2012.

[Be09a] Becker, Jörg; Delfmann, Patrick; Herwig, Sebastian; Lis, Łukasz; Stein, Armin: Formalizing linguistic conventions for conceptual models. In: Conceptual Modeling-ER 2009, pp. 70–83. Springer, 2009.

[Be09b] Becker, Jörg; Delfmann, Patrick; Herwig, Sebastian; Lis, Lukasz; Stein, Armin: Towards increased comparability of conceptual models-enforcing naming conventions through domain thesauri and linguistic grammars. pp. 2231–2242, 2009.

[BK04] Berry, Daniel M.; Kamsties, Erik: Ambiguity in Requirements Specification. In: Perspectives on Software Requirements, volume 753 of The Springer International Series in Engineering and Computer Science, pp. 7–44. Springer, 2004.

[Co12] Combi, Carlo; Gambini, Mauro; Migliorini, Sara; Posenato, Roberto: Modelling temporal, data-centric medical processes. In: ACM International Health Informatics Symposium. pp. 141–150, 2012.

[Da92] Dalianis, Hercules: A method for validating a conceptual model by natural language discourse generation. In: Advanced Information Systems Engineering. Springer, pp. 425–444, 1992.

[dd13] de Almeida Ferreira, David; da Silva, Alberto Rodrigues: RSL-PL: A linguistic pattern language for documenting software requirements. In: Third IEEE International Workshop on Requirements Patterns. pp. 17–24, 2013.

[Di08] Dixon, Robert MW: Deriving verbs in English. Language Sciences, 30(1):31–52, 2008.

[EKO07] Ehrig, Marc; Koschmider, Agnes; Oberweis, Andreas: Measuring similarity between semantic business process models. In: Proceedings of the fourth Asia-Pacific conference on Comceptual modelling-Volume 67. pp. 71–80, 2007.

[Fe98] Fellbaum, Christiane, ed. WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press, May 1998.

[FMP11] Friedrich, Fabian; Mendling, Jan; Puhlmann, Frank: Process model generation from natural language text. In: Advanced Information Systems Engineering. Springer, pp. 482–496, 2011.

[GL11] Gruhn, Volker; Laue, Ralf: Detecting Common Errors in Event-Driven Process Chains by Label Analysis. Enterprise Modelling and Information Systems Architectures, 6(1):3–15, 2011.

[GSD99] Gomez, Fernando; Segami, Carlos; Delaune, Carl: A system for the semiautomatic generation of ER models from natural language specifications. Data & Knowledge Engineering, 29(1):57–81, 1999.

[HD15] Höhenberger, Steffen; Delfmann, Patrick: Supporting Business Process Improvement through Business Process Weakness Pattern Collections. In: 12. Internationale Tagung Wirtschaftsinformatik. pp. 378–392, 2015.

[HWPZ03] Heemskerk, Marieke; Wilson, Karen; Pavao-Zuckerman, Mitchell: Conceptual models as tools for communication across disciplines. Conservation Ecology, 7(3):8, 2003.

[KHO11] Koschmider, Agnes; Hornung, Thomas; Oberweis, Andreas: Recommendation-based editor for business process modeling. Data & Knowledge Engineering, 70(6):483–503, 2011.

[Le13] Leopold, Henrik; Eid-Sabbagh, Rami-Habib; Mendling, Jan; Azevedo, Leonardo Guerreiro; Baião, Fernanda Araujo: Detection of naming convention violations in process models for different languages. Decision Support Systems, 56:310–325, 2013.

[Li01] Liddy, Elizabeth D: Natural language processing. 2001.

[LRR96] Lavoie, Benoit; Rambow, Owen; Reiter, Ehud: The modelexplainer. In: Proceedings of the 8th international workshop on natural language generation. pp. 9–12, 1996.

[LSM09] Leopold, Henrik; Smirnov, Sergey; Mendling, Jan: On labeling quality in business process models. Nüttgens M et al.(Hrsg), 8:42–57, 2009.

[LSM11] Leopold, Henrik; Smirnov, Sergey; Mendling, Jan: Recognising Activity Labeling Styles in Business Process Models. Enterprise Modelling and Information Systems Architectures, 6(1):16–29, 2011.

[LSM12] Leopold, Henrik; Smirnov, Sergey; Mendling, Jan: On the refactoring of activity labels in business process models. Information Systems, 37(5):443–459, 2012.

[LSS94] Lindland, Odd Ivar; Sindre, Guttorm; Solvberg, Arne: Understanding quality in conceptual modeling. Software, IEEE, 11(2):42–49, 1994.

[MAA08] Meziane, Farid; Athanasakis, Nikos; Ananiadou, Sophia: Generating Natural Language specifications from UML class diagrams. Requirements Engineering, 13(1):1–18, 2008.

[Me02] Megyesi, Beáta: Shallow parsing with PoS taggers and linguistic features. The Journal of Machine Learning Research, 2:639–668, 2002.

[Mo08] Montes, Azucena; Pacheco, Hasdai; Estrada, Hugo; Pastor, Oscar: Conceptual model generation from requirements model: A natural language processing approach. In: Natural Language and Information Systems, pp. 325–326. Springer, 2008.

[MRR10] Mendling, Jan; Reijers, Hajo A; Recker, Jan: Activity labeling in process modeling: Empirical insights and recommendations. Information Systems, 35(4):467–482, 2010.

[NH15] Niesen, Tim; Houy, Constantin: Zur Nutzung von Techniken der Natürlichen Sprachverarbeitung für die Bestimmung von Prozessmodellähnlichkeiten–Review und Konzeptentwicklung. In: 12. Internationale Tagung Wirtschaftsinformatik, WI 2015, Osnabrück, Germany, March 4-6, 2015. pp. 913–924, 2015.

[Sa90] Santorini, Beatrice: Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). 1990.

[Si11] Silver, Bruce: BPMN method and style. Cody-Cassidy Press, US, 2nd ed. edition, 2011.

[SLG13] Storch, Arian; Laue, Ralf; Gruhn, Volker: Measuring and visualising the quality of models. In: IEEE International Workshop on Communicating Business Process and Software Models Quality. IEEE, pp. 1–8, 2013.

[SLG14] Storch, Arian; Laue, Ralf; Gruhn, Volker: Analysing the Style of Textual Labels in i* Models. In: International i* Workshop co-located with the 26th International Conference on Advanced Information Systems Engineering (CAiSE 2014). 2014.

[To03] Toutanova, Kristina; Klein, Dan; Manning, Christopher D; Singer, Yoram: Feature-rich part-of-speech tagging with a cyclic dependency network. In: 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology. pp. 173–180, 2003.

[vdVGvdR97] van der Vos, Bram; Gulla, Jon Atle; van de Riet, Reind: Verification of conceptual models based on linguistic knowledge. Data & knowledge engineering, 21(2):147–163, 1997.