MATHEMATICAL LOGIC PROBLEMS IN NATURAL LANGUAGE PROCESSING

Serhii Kryvyi, Nataliia Darchuk, Hryhorii Hoherchak

***Abstract**: The article describes some ways of knowledge bases application to natural language texts analysis and solving some of their processing tasks. The basic problems of natural language processing are considered, which are the basis for their semantic analysis: problems of tokenization, parts of speech tagging, dependency parsing, correference resolution. The basic concepts of knowledge bases theory are presented and the approach to their filling based on Universal Dependencies framework and the correference resolution problem is proposed. Examples of applications for knowledge bases filled with natural language texts in practical problems are given, including checking constructed syntactic and semantic models for consistency and question answering.*

***Keywords**: knowledge bases, natural language processing, syntax dependencies, coreference resolution, semantic analysis.*

***ITHEA Keywords**: F.4.1 Mathematical Logic; I.2.4 Knowledge Representation Formalisms and Methods; I.2.7 Natural Language Processing.*

Introduction

The history of the natural language processing industry generally dates to the 1950s. The first task for scientists was the task of automating translation: for the United States Government, it was important to have a system that would allow translating Russian-language texts into English with high accuracy. As early as 1954, the primitive machine translation system was first demonstrated within the Georgetown experiment.

The modern field of natural language processing has more than three dozen basic tasks, including tasks of part of speech tagging, text tokenization, dependency parsing, coreference resolution, lexical normalization, named entities recognition, search of the missing components, natural language inference, speech recognition and understanding, machine translation, sentiment analysis, grammar checking and more. The current status of a large number of these and similar tasks is described on the NLP-progress portal and on the Natural Language Processing page of Papers With Code. Last one provides ratings of models for solving each of these problems with links to scientific articles describing corresponding machine learning models, as well as, for many of those, references to the source codes of the respective machine learning models.

Such tasks include open information extraction (OpenIE), the purpose of which is to present natural-language text in a structured form: usually in the form of binary or larger dimension relations. A qualitative solution to this problem would give an opportunity, to some extent, to talk about the presence of automated methods of filling the natural-language knowledge base, the content of which consists of atomic concepts and roles - relations between them. Mathematical formulation of this problem was also proposed in [Palagin et al, 2012].

At the time of writing this article, this task does not have clearly formulated and generally accepted standards of what is considered a qualitative result: that is, what relationships should be obtained and how exactly they should be written. Thus, there are also no standards of evaluation for such models and corpora of acceptable size for the qualitative training of machine learning models, as it is accepted for many of the tasks mentioned above.

The first steps towards the specification and evaluation of the results of this problem were made in [Stanovsky and Dagan, 2016], which offers a comparison of OpenIE models based on the precision-recall curve and the AUC (area under the curve) metric. Most of the new models use the evaluation technique proposed in [Cetto et al, 2018], although some new works offer comparisons based on the more common F1 metric [Zhan and Zhao, 2019] [Léchelle, 2019].

In general, the models proposed for this task are divided into two subtypes [Niklaus, 2018]:

- machine-based systems (e.g. Neural Open Information Extraction, OpenIE-5.0);
- rule-based systems (e.g. Graphene [Cetto et al, 2018]).

It should be noted that the quality of modern models for this task (even measured by existing F1 and AUC metrics) does not allow to speak about the qualitative construction of knowledge bases based on natural language text information at this stage. Furthermore, the problem comes even worse solved, if we view the variety of problem statements and metrics to compare its results.

Thus, extraction of open (arbitrary) relations from natural-language texts is quite promising research topic. Open questions include formalization of the OpenIE problem in view of its application in the natural language knowledge base filling, along with the construction of a metric apparatus for comparing models, and, finally, solving the task.

Construction of a natural language knowledge base enables one to analyze the text properties using algorithms and methods of description logic for knowledge bases. Using the algorithm for validating concepts based on semantic tables in field of a text-based knowledge base allows one to test the consistency of a syntactic and semantic model built by any models. Thus, with some additional knowledge of the subject area, it is possible to identify contradictory, and therefore erroneous, elements in order to correct them in future. The partially solved problem of answer queries to the knowledge base, when converting a text question into an appropriate query language expression, is also a useful mean of solving the question answering task.

Applied natural language processing tasks overview

Potentially useful inputs for the problem of open information extraction can be gained from the results of text analysis for parts of speech, named entities, grammatical dependencies and coreferences.

Tokenization problem

The task of tokenization in the field of natural language processing aims to process a sequence of characters (text) and to split it into individual words or sentences. Words extraction, as the first approximation, can be done by splitting the input character stream into parts by separators (for example, spaces, punctuation marks). Although, full tokenization should also take into account the features of certain languages, where punctuation may be a part of complex lexical constructions (for example, in English, the sequence of characters *i.e.* corresponds to the phrase *in other words*, and the construction *let's* stands for two words: *let and us*) or abbreviations. Similarly, it is not entirely possible to break down the text into delimiters and to divide it into sentences, since punctuation marks, as noted above, can also be part of words and complex speech constructions.

Since, as noted above, the tokenization algorithms consider language features, tokenization algorithms are usually constructed for each language or group of similar languages separately. For example, the separation of English-language texts into words and sentences can be carried out using the Stanford Tokenizer proposed in [Manning et al, 2002].

Part of speech tagging

The task of POS-tagging is to label each word in the text as part of the speech of the language, which it belongs to. Modern works in the field of natural language processing mostly use morphological designations defined in Universal Dependencies [McDonald et al, 2003], a framework for a single annotation system for grammars of different natural languages. This framework allows you to work with the morphological and grammatical structure of the sentence, paying no (or very little) attention to peculiarity of each specific language and operating only with appropriate universal designations.

Consider the following sentence:

Oral messages are recorded on paper, replacing the sounds of human language with the letters of the alphabet.

The corresponding result of the Universal Dependencies morphological analysis is presented in Figure 1.

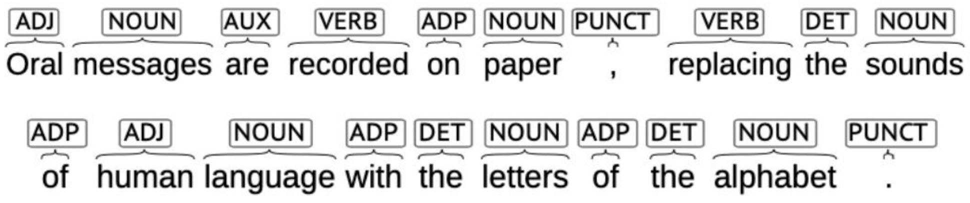

ADJ NOUN AUX VERB ADP NOUN PUNCT VERB DET NOUN
Oral messages are recorded on paper , replacing the sounds
ADP ADJ NOUN ADP DET NOUN ADP DET NOUN PUNCT
of human language with the letters of the alphabet .

Figure 1. Results of part of speech tagging for English sentence

Here we use *ADJ* for adjectives, *NOUN* for nouns, *VERB* for verbs, *ADP* for prepositions and *PUNCT* for punctuation marks.

Modern models of part of speech tagging are mostly based on a machine learning approach. They usually use the standard dataset – part of the Penn Treebank, that corresponds to the Wall Street Journal, which contains 45 different POS tags to accomplish this task in English. The best precision at the time of writing this text at 97.96% is demonstrated by the Meta BiLSTM model proposed in [Bohnet, 2018]. This model is based on two recurrent neural networks with context at the sentence level, the results of which are combined using a meta-model so that the output produces a unified representation of each word, which is then used to refer.

Solving a similar problem for many languages at the same time, using tags from the Universal Dependencies framework and corresponding corpora for different languages, is more difficult. Currently, several models, including Uppsala and HIT-SCIR, perform better for a large number of languages (the average F1 score across all languages for both models exceeds 0.9 for this task). In particular, the HIT-SCIR and Stanford models reach the F1 score above 0.97 for English, Ukrainian and Russian.

Dependency parsing

The task of dependency parsing is to identify the dependencies that represent grammatical structure of the given sentence and determine the links between the parental words and the words that modify them (i.e. child).

Common dependency notion principles are also provided in Universal Dependencies framework, which defines more than 30 different types of dependencies and some extensions for them, specific for some limited subset of supported languages. In the basic version of these dependencies, the syntactic structure of the sentence is presented in the form of a tree, that is, each word of the sentence (except the main one - the root) has exactly one ancestor. Each branch of the tree is marked with a special tag that categorizes the relationship between the ancestor word and the descendant word by one of 36 different types of dependencies.

An example of the parsing result of the above-mentioned sentence is given in Figure 2.

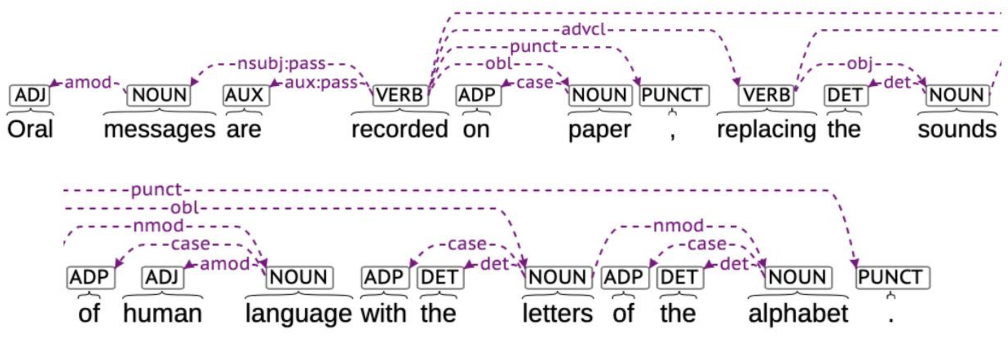


Figure 2. Results of dependency parsing for English sentence

Here *amod* denotes an adjectival modifier, *obj* – an object, *obl* – a circumstance of the action, *advcl* – an adverbial-participial turnover, *case* – an auxiliary word, *nmod* – a noun modifier and *punct* – a punctuation mark.

Models for solving this problem in English are mostly compared on the basis of the Penn Treebank dataset with the predicted designations of parts of the speech. The following metrics are used to compare them:

– UAS (unlabeled attachment score), which does not take into account the dependency mark, but compares only the correct ancestor definition of each word;

– LAS (labeled attachment score), which denotes the proportion of correctly parsed words (correctly defined ancestors and dependency marks).

At the time of writing of this article, the Label Attention Layer + HPSG + XLNet model, proposed in November 2019 [Mrini, 2019], demonstrates the best evaluation results. This model is also based on the machine learning approach, reaching UAS 97.33% and LAS 96.29%. However, recent scientific conferences have focused on building unified parsing models for many languages. Thus, the HIT-SCIR model achieves LAS of 0.92, 0.88 and 0.87 for Russian, Ukrainian and English respectively.

Consider a more complex sentence: *Cats usually catch and eat mice and rats*.

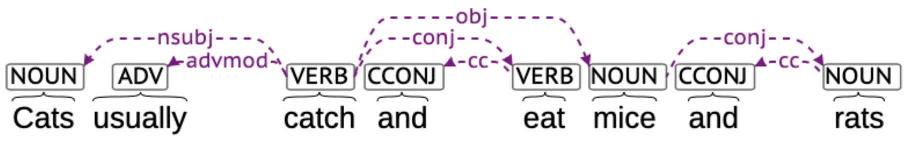


Figure 3. Results of dependency parsing for English sentence

Dependency tree from Figure 3 shows that basic dependencies do not sufficiently describe semantic links. Here action objects *mouse* and *rat* are related by the conjunction relationship, so that the word *rat* is associated with the action only indirectly. Although semantically both are still the objects of the *catch* action.

Similarly, actions catch and eat are conjunct in this sentence, too. But the above-mentioned dependency tree does not show subjects nor objects of the second action straightforward.

These and other problems are solved by expanding the dependency tree with additional arcs (Figure 4) – in cost of the loss of its exact tree structure. Converting a basic dependency tree to an extended dependency graph requires solution of the following problems:

- elided predicate restoration by creating null nodes;
- propagation of conjuncts (objects, entities, definitions);
- subject distribution to subordinate verbs of complex predicate;
- additional processing of relative clauses (may lead to cycles);
- adding case information to the dependency name.

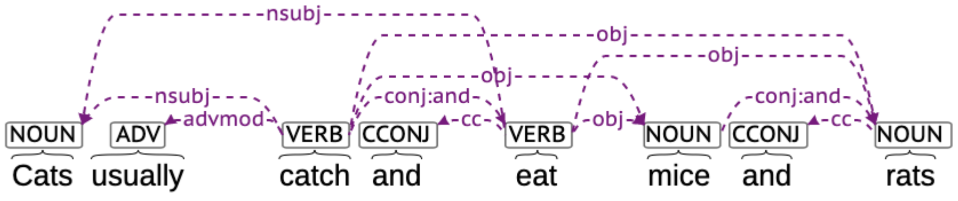


Figure 4. Enhanced dependencies graph

CoreNLP natural language package delivers 0.92 for the English LAS. The presence of a corpus for Ukrainian language allow us to speak about the potential for solving this problem for this language, but no models were openly available. No publicly available models for Universal enhanced dependencies were found while writing this article, too.

Along with Universal dependencies, there are also several specialized, mostly language-specific, dependency formats. An example of an alternative format for describing the syntactic structure of a sentence in the Ukrainian language proposed in [Darchuk, 2013] is given in comparison with the universal dependencies in Figures 5, 6.

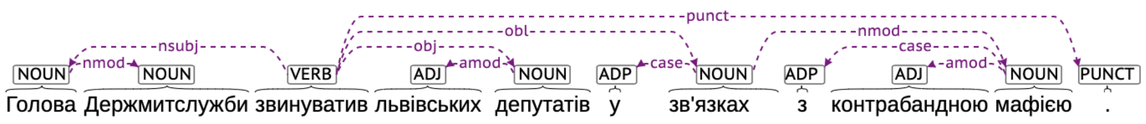


Figure 5. Example of Universal dependencies tree for a sentence in Ukrainian

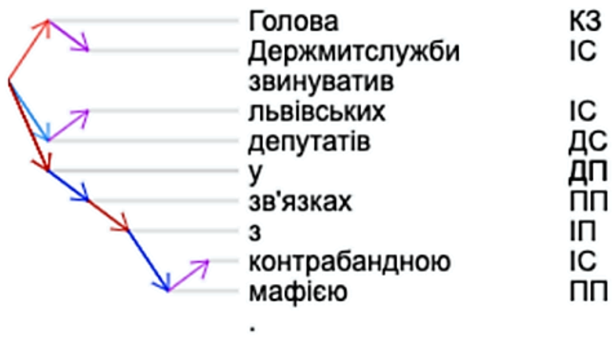


Figure 6. Example of mova.info dependencies tree for a sentence in Ukrainian

Here, the symbol *КЗ* denotes the subject and predicate compound, the *ІС* is the non-prepositional noun compound, the *ДС* is the non-prepositional verb compound, *ДП* is the prepositional verb compound, *ПП* is the preposition compound, and *ІП* is the prepositional noun compound.

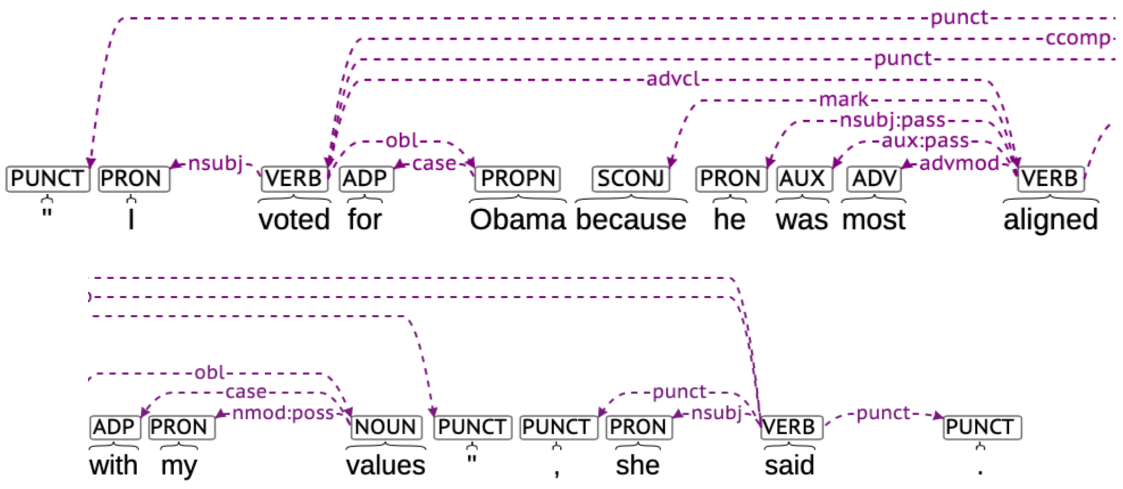
Combining the results of different parsing formats allows one to achieve a better aggregate result and to correct errors that occurred in each of the resulting trees.

Coreference resolution

The coreference resolution task aims to cluster the references in the text that relate to the same entity of the real world.

Consider the following sentence: *"I voted for Barack Obama, because his beliefs are closest to my own values," she said.*

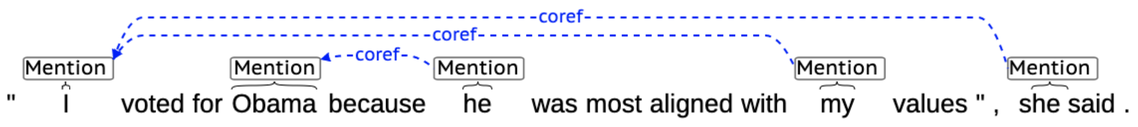
Analysis of its syntactic dependencies (Figure 7) does not allow to fully determine which objects of the real world – identical or different, – refer to the pronouns that appear in the sentence. A similar problem, but extended to mentions throughout the text, not just within a sentence, cannot be solved at all with either basic or extended dependencies, since they represent links within only one sentence.



punct
ccomp
punct
advcl
mark
nsubj:pass
aux:pass
obl
nsubj
case
advmod
PUNCT PRON VERB ADP PROPN SCONJ PRON AUX ADV VERB
" I voted for Obama because he was most aligned
obl
case
nmod:poss
punct
nsubj
punct
ADP PRON NOUN PUNCT PUNCT PRON VERB PUNCT
with my values " , she said .

Figure 7. Dependency tree with many coreferential words and phrases

The foregoing causes the emergence of a separate problem of natural language processing, the solution of which would allow to gather equivalent entities in a text into one and analyze all contained relations more fully.



coref
coref
coref
Mention Mention Mention Mention Mention
" I voted for Obama because he was most aligned with my values " , she said .

Figure 8. Coreference trees forest

The set of coreferential words and phrases is usually represented in the form of a forest (Figure 8) – sets of trees, each of which denotes a set of reference nodes. The arc of reference is usually directed to the most specific designation of a real-world object.

Models are currently being evaluated on OntoNotes, which contains English texts (news, phone calls, blogs, talk shows, etc.) with pre-annotated coreferences.

Currently, the state-of-the-art models are mostly modifications to the BERT model [Devlin et al, 2019], which is based on a machine learning approach and was developed by the Google AI Language team. BERT offers a common model for the presentation of natural language information for a range of word processing tasks and considers context on both directions, as opposed to using left- or right-handed contexts in previous efficient models.

Basic concepts of knowledge base theory

We introduce some key basics of the theory of knowledge bases and description logics that will be used below in this article.

Concepts are a tool for recording knowledge about the subject area to which they apply. This knowledge is divided into general knowledge of concepts and their interconnections and knowledge of individual objects, their properties and relations with other objects. According to this division, knowledge written using the language of description logic is divided into a set of terminal axioms named TBox and a set of facts about individuals named ABox.

Let $CN = \{A_1, ..., A_n\}$ and $RN = \{R_1, ..., R_m\}$ – finite non-empty sets of *atomic concepts (concept names)* and *atomic roles (role names)* accordingly. [Baader et al, 2007]

Definition 1. A set of concepts of $\mathcal{ALC}$-logic is defined as follows:

– *symbols $\top$ (top) and $\bot$ (bottom) are concepts;*

– *each $A_i \in CN$ is a concept;*

– *if C is a concept then $\neg C$ $(\text{complement of } C)$ is a concept;*

– *if C and D are concepts then $C \sqcap D$ (intersection of concepts) and $C \sqcup D$ (union of concepts) are concepts;*

- *if C is a concept and R is an atomic role then $\exists R.C$ and $\forall R.C$ are concepts;*
- *no other expressions are concepts.*

Definition 2. Interpretation is a pair $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ that consists of a non-empty set Δ named domain and interpretation function $\cdot^{\mathcal{I}}$ that maps:

- *to each atomic concept $A \in CN$ an arbitrary subset $A^{\mathcal{I}} \subseteq \Delta$;*
- *to each atomic role $R \in RN$ an arbitrary subset $R^{\mathcal{I}} \subseteq \Delta \times \Delta$.*

Interpretation function is being spread on the whole set of $\mathcal{ALC}$-logic concepts unambiguously:

- $\top^{\mathcal{I}} = \Delta$; $\bot^{\mathcal{I}} = \varnothing$; $(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$; $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- $(\exists R.C)^{\mathcal{I}} = \{e \in \Delta \mid \exists d \in \Delta : (e,d) \in R^{\mathcal{I}} \wedge d \in C^{\mathcal{I}}\}$;
- $(\forall R.C)^{\mathcal{I}} = \{e \in \Delta \mid \exists d \in \Delta : (e,d) \in R^{\mathcal{I}} \rightarrow d \in C^{\mathcal{I}}\}$.

Definition 3. A terminological axiom is an expression $C \sqsubseteq D$ (inclusion of a concept C into a concept D) or $C \equiv D$ (equivalence of concepts C and D), where C and D are arbitrary concepts.

Terminology (TBox) is an arbitrary finite set of terminological axioms.

Definition 4. An axiom $C \sqsubseteq D$ ($C \equiv D$) is true in interpretation I if $C^I \subseteq D^I$ ($C^I = D^I$). In this case I is called a model of this axiom and write $I \vDash C \sqsubseteq D$. An interpretation I is called a model of terminology T ($I \vDash T$) if it is a model for all axioms of.

Terminology is called compatible or executable if it has a non-empty model. The concept C is executed with respect to terminology T if there is a model I of terminology T such that $C^I \neq \varnothing$.

The terminology gives the opportunity to record general knowledge of concepts and roles. But it often insinuates the need to record knowledge about specific individuals: what class the individual belongs to, what relationships (roles) they relate to each other.

Definition 5. A system of facts (ABox) is a finite set A of statements of a kind $a : C$ or aRb, where a,b are individuals, C – an arbitrary concept and R – a role.

Below we will return to the concept of executability of terminology and concept and provide the tableau algorithm for this problem.

$\mathcal{ALC}$ -logic described above can be extended by adding new concept constructors to its syntax definition.

Yes, $\mathcal{ALCQ}$-logic can be built on $\mathcal{ALC}$-logic with addition of rules as follows:

– *if C is a concept, R is an atomic role and n is a natural number then $\leq nR.C$ is a concept.*

Semantics of these kinds of roles are the following:

– $(\leq nR.C)^{\mathcal{I}} = \{e \in \Delta \mid \left|\{d \in C^{\mathcal{I}} : (e,d) \in R^{\mathcal{I}}\}\right| \leq n\}$.

Naturally, we can define the following notions:

– $\geq nR.C = \neg(\leq (n-1)R.C)$;

– $= nR.C = (\leq nR.C) \sqcap (\geq nR.C)$;

– $< nR.C = (\leq nR.C) \sqcap \neg(= nR.C)$;

– $> nR.C = (\geq nR.C) \sqcap \neg(= nR.C)$.

Also, we can define $\exists R.C = (\geq 1R.C)$.

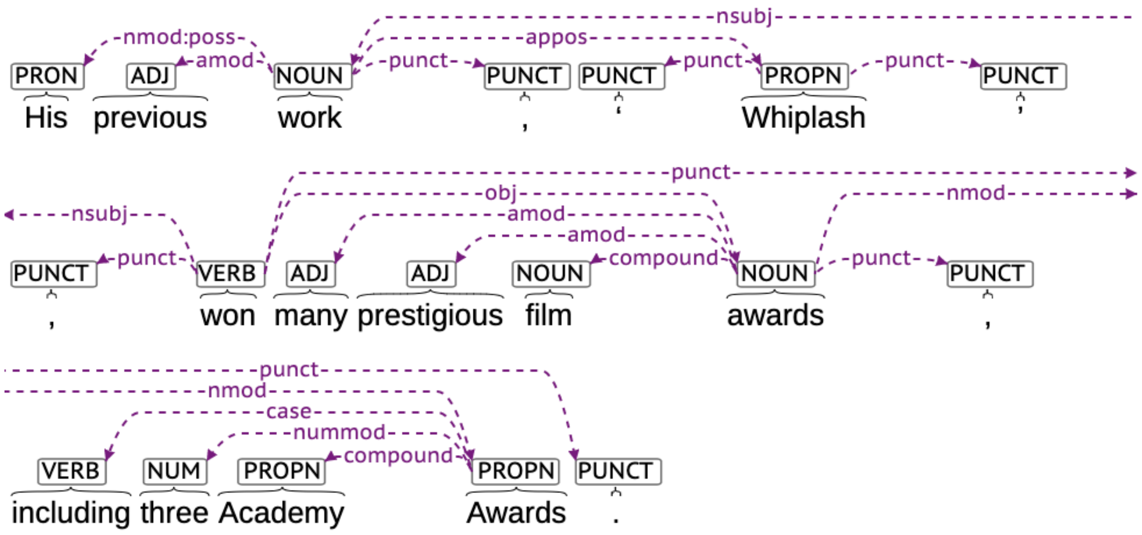
Dependency-based knowledge base population

The tree (or graph) of syntactic dependencies discussed above is a powerful source for extracting knowledge in the form of open relations. Consider the following text:

‘La La Land’ is the third film by young director Damien Chazelle. His previous work, ‘Whiplash’, won many prestigious film awards, including three Academy

Awards. This year nominees are already known, and the 'La La Land' is the undisputed leader: has 14 nominations. This picture has already won all the most prestigious nominations of Golden Globe Awards.

The main sources of relations, that is, the facts of a kind aRb, are the verbs along with the words, connected by *nsubj* (nominal subject) and *obj* (object) dependencies. Consider the second sentence of the above text, the dependency tree for which is given in Figure 9.



PRON ADJ NOUN PUNCT PUNCT PROPN PUNCT
His previous work , ' Whiplash '
nmod:poss
amod
nsubj
appos
punct
PUNCT VERB ADJ ADJ NOUN NOUN PUNCT
, won many prestigious film awards ,
nsubj
punct
obj
amod
compound
nmod
VERB NUM PROPN PROPN PUNCT
including three Academy Awards .
case
nummod

Figure 9. Dependency tree

From this dependency tree, we can extract a relation triple *(work; win; film award)*. Obviously, such a relation itself does not carry enough substantive load – all that because the individuals involved here need additional specification.

Using *flat* and *amod* dependencies, let's construct the following chain of concepts for the subject of this action:

$$work\_previous \sqsubseteq work,$$

$$work\_Whiplash \sqsubseteq work\_previous$$

The fact that individual a_1 belongs to the generated concepts can be written as: a_1:*work_Whiplash*. We shall note that the affiliation of this individual to other concepts follows from the essence of the concept inclusion relation.

In the same way for the object of the action we can write the following terminological axiom:

$$\mathit{film\_award\_prestigious} \sqsubseteq \mathit{film\_award}$$

On this stage we shall notice that here we will pay reader's attention only to population of a TBox. ABox facts can be constructed in accordance to the TBox terminology considered below by deterministic algorithm of semantic table, which will be considered below.

Since the object is in plural, the following concepts should be included:

$$\mathit{work\_Whiplash} \sqsubseteq \ \geq 2R_{win}.\mathit{film\_award\_prestigious}$$

Another R_{win} role subject is hidden in the basic dependency tree behind *conj* dependency. After similar operations, we can obtain the following knowledge base:

$$\mathit{TBox} = \{\, \mathit{work\_previous} \sqsubseteq \mathit{work},\ \mathit{work\_Whiplash} \sqsubseteq \mathit{work\_previous},$$
$$\mathit{film\_award\_prestigious} \sqsubseteq \mathit{film\_award},$$
$$\mathit{work\_Whiplash} \sqsubseteq \ \geq 2R_{win}.\mathit{film\_award\_prestigious},\ \mathit{Academy\_Award} \sqsubseteq \mathit{award},$$
$$\mathit{work\_Whiplash} \sqsubseteq \ = 3R_{win}.\ \mathit{Academy\_Award}\,\}$$

Except *flat*, *nmod*, and *amod* dependencies, new terminological axioms can also be formed from *obj* dependencies in the case of an elided predicate. Yes, in the first sentence of the text above (Figure 10) the verb *is* is considered as a copula.

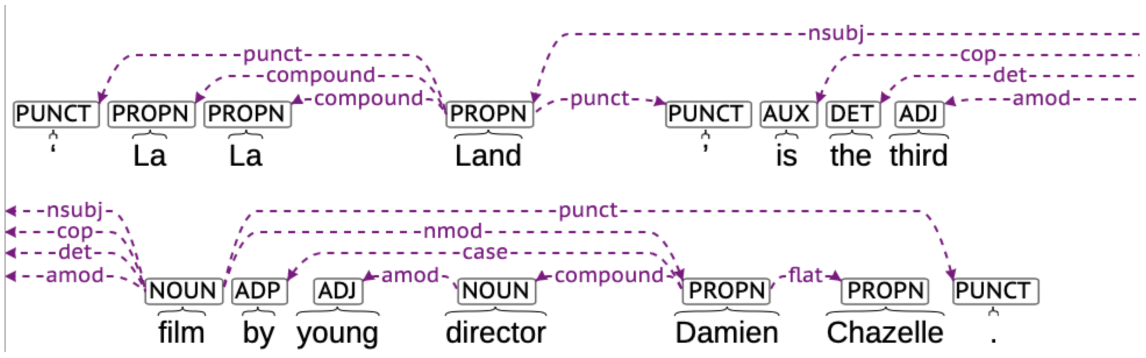


Figure 10. Dependency tree

Thus, since the dependency tree root is a noun, dependencies *obj* and *subj* here semantically denote the inclusion of concepts. Similarly, we produce concepts and terminological axioms for object and the root of this tree as follows:

$$director\_Damien\_Chazelle\_film \sqsubseteq film,$$
$$young\_director\_Damien\_Chazelle\_film \sqsubseteq director\_Damien\_Chazelle\_film,$$

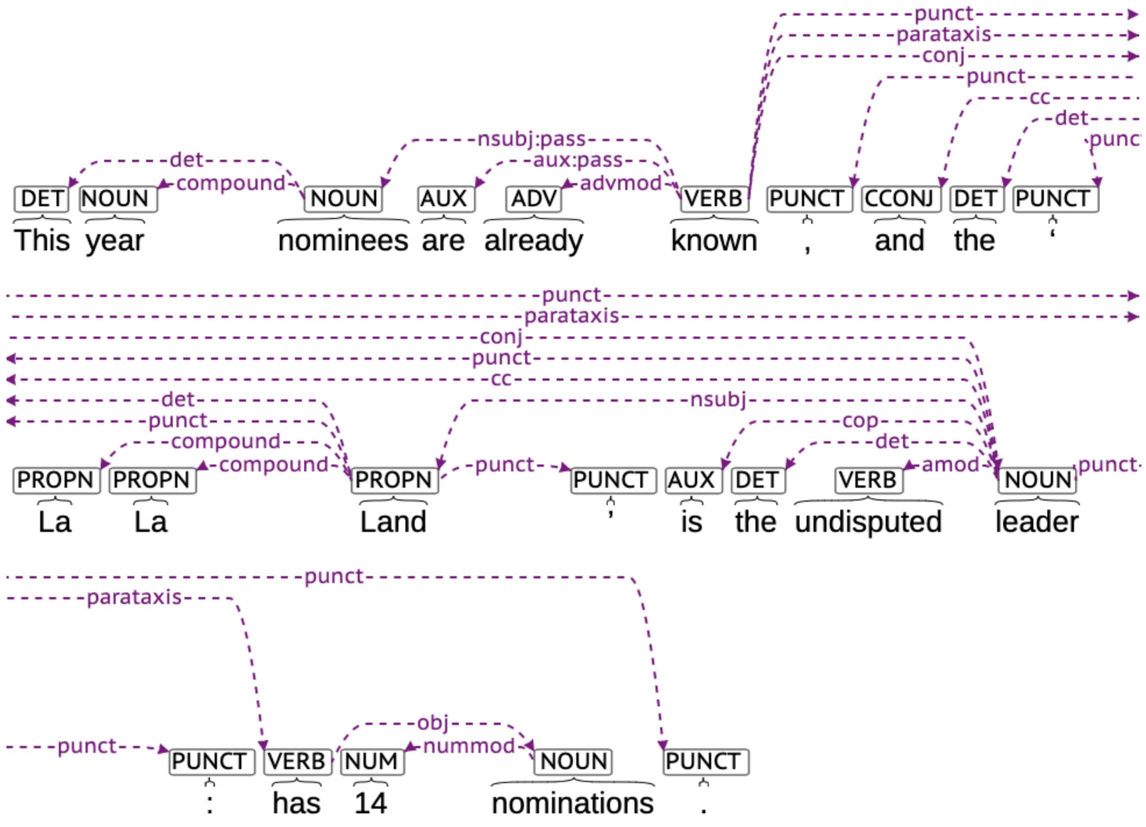
$$young\_director\_Damien\_Chazelle\_film\_third \sqsubseteq$$
$$young\_director\_Damien\_Chazelle\_film$$

In addition, for the considered sentence, the following axiom will be added to the list of terminological axioms:

$$La\_La\_Land \sqsubseteq young\_director\_Damien\_Chazelle\_film\_third$$

In cases when the root of a sentence is an adjective or a past participle, it also means an inclusion of concepts. Yes, the first part of the third sentence (Figure 11) produces the following terminological axioms:

this_year_nominee $\sqsubseteq$ *nominee, this_year_nominee* $\sqsubseteq$ *known*



This year nominees are already known , and the ‘
DET NOUN NOUN AUX ADV VERB PUNCT CCONJ DET PUNCT
det compound nsubj:pass aux:pass advmod punct parataxis conj punct cc det punct
La La Land ’ is the undisputed leader
PROPN PROPN PROPN PUNCT AUX DET VERB NOUN
compound punct det cc conj nsubj cop amod
: has 14 nominations .
PUNCT VERB NUM NOUN PUNCT
obj nummod parataxis punct

Figure 11. Dependency tree

In cases when the root of a sentence has *conj* dependents associated with it, a each such dependent is considered and processed as the root of its subtree. Since the root of the subtree is a noun, like the case considered earlier, the knowledge base is supplemented by the following terminological axioms:

undisputed_leader $\sqsubseteq$ *leader, La_La_Land* $\sqsubseteq$ *undisputed_leader*

Parataxis-dependent subtrees should be treated in the very same way. So, in this sentence, as a separate statement, we consider the expression has 14 nominations, which adds to the knowledge base only the concept nomination. Since the verb has no subject, there is a problem of defining it with context. In this case, the definition is simple enough: just use the subject of the ancestor, that is, the tape "La La Land". Thus, in addition to the knowledge base, the following facts should be added:

Parataxis-dependent subtrees should be treated in the same way. Thus, consider the expression *has 14 nominations*, which adds to the knowledge base the only concept *nomination*. Since the predicate has no subject, here comes a problem of defining it within context. In this case, the definition is simple enough: we use the subject of the ancestor, that is, the phrase *La La Land*. Thus, the following axiom should be added to the knowledge base:

$$La\_La\_Land \sqsubseteq R_{has}.nomination$$

This terminological axiom can be clarified with numerical modifier mentioned in the sentence:

$$La\_La\_Land \sqsubseteq\ =14R_{has}.nomination$$

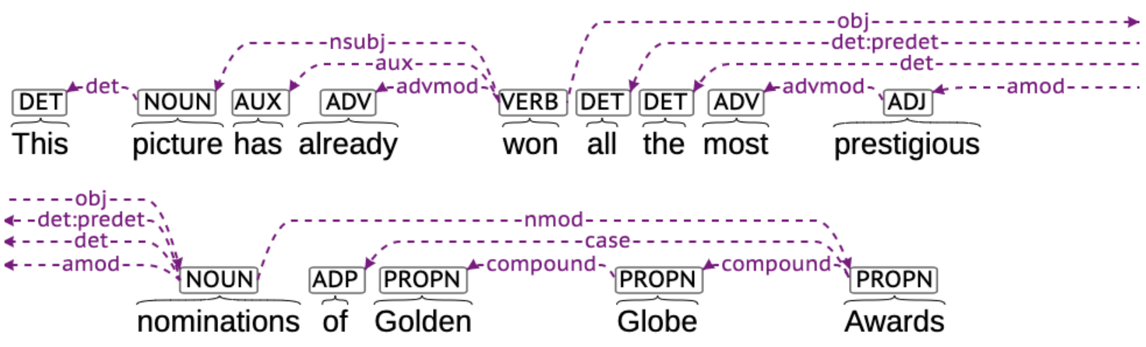


Figure 12. Dependency tree

Last sentence (Figure 12) will produce the following terminological axioms:

Golden_Globe_Awards_nomination $\sqsubseteq$ *nomination,*
Golden_Globe_Awards_nomination_prestigious $\sqsubseteq$
Golden_Globe_Awards_nomination,
Golden_Globe_Awards_nomination_most_prestigious $\sqsubseteq$
Golden_Globe_Awards_nomination_prestigious,

Considering coreference resolution task as solved, the subject *This picture* corresponds to the concept *La_La_Land*. Thus, the knowledge base is also supplemented by the following axiom:

La_La_Land $\sqsubseteq R_{win}$ *.Golden_Globe_Awards_nomination_most_prestigious*

The final knowledge base for the given text fragment will look like this:

CN = {*work, work_previous*, *work_Whiplash*, *film_award*,
film_award_prestigious, *Academy_Award, award, film*,
director_Damien_Chazelle_film, *young_director_Damien_Chazelle_film*,
young_director_Damien_Chazelle_film_third, *La_La_Land, nominee*,
this_year_nominee, *known*, *leader*, *undisputed_leader*, *nomination*,
Golden_Globe_Awards_nomination,
Golden_Globe_Awards_nomination_prestigious,
Golden_Globe_Awards_nomination_most_prestigious}

RN = $\{R_{win}, R_{has}\}$

TBox = {*work_previous* $\sqsubseteq$ *work, work_Whiplash* $\sqsubseteq$ *work_previous,*
film_award_prestigious $\sqsubseteq$ *film_award,*
work_Whiplash $\sqsubseteq \geq 2R_{win}$*.film_award_prestigious, Academy_Award* $\sqsubseteq$ *award,*

work_Whiplash $\sqsubseteq \; =3R_{win}$. *Academy_Award,*

director_Damien_Chazelle_film $\sqsubseteq$ *film,*

young_director_Damien_Chazelle_film $\sqsubseteq$ *director_Damien_Chazelle_film,*

young_director_Damien_Chazelle_film_third $\sqsubseteq$

young_director_Damien_Chazelle_film,

La_La_Land $\sqsubseteq$ *young_director_Damien_Chazelle_film_third*,

this_year_nominee $\sqsubseteq$ *nominee, this_year_nominee* $\sqsubseteq$ *known*,

undisputed_leader $\sqsubseteq$ *leader, La_La_Land* $\sqsubseteq$ *undisputed_leader*,

La_La_Land $\sqsubseteq \; =14R_{has}$*.nomination*,

Golden_Globe_Awards_nomination $\sqsubseteq$ *nomination,*

Golden_Globe_Awards_nomination_prestigious $\sqsubseteq$

Golden_Globe_Awards_nomination,

Golden_Globe_Awards_nomination_most_prestigious $\sqsubseteq$

Golden_Globe_Awards_nomination_prestigious,

La_La_Land $\sqsubseteq \exists R_{win}$ *.Golden_Globe_Awards_nomination_most_prestigious* }

Usage of WordNet's lexical database during semantic analysis one can also add the following auxiliary terminological axioms:

{*film* $\sqsubseteq$ *work, film_award* $\sqsubseteq$ *award*}

Tableau algorithm

Definition 6. Algorithm U *solves the concept execution problem in terminology* T *for description logic* L *if the following conditions are satisfied:*

- *Finiteness: for arbitrary* (C,T) *algorithm* U *generates answer* $U(C,T)$ *in a finite time;*

– *Correctness: for arbitrary* (C,T) *if* C *is executable in terminology* T *then* $U(C,T)=1$.

– *Completeness: for arbitrary* (C,T) *if* $U(C,T)=1$ *then concept* C *is executable in terminology* T *.*

The tableau algorithm for concept execution checking is defined by the rules in Tables 1.

Table 1. Rules of $\mathcal{ALC}$ ***+TBox tableau algorithm***

Rule	Conditions of applying	Action
$\sqcap$-rule	point x is active; $x:(C \sqcap D) \in \mathcal{A}$	$\mathcal{A}' = \mathcal{A} \cup \{x:C, x:D\}$
$\sqcup$-rule	point x is active; $x:(C \sqcup D) \in \mathcal{A}$	$\mathcal{A}' = \mathcal{A} \cup \{x:C\}$, $\mathcal{A}'' = \mathcal{A} \cup \{x:D\}$
$\exists$-rule	point x is active; $x:\exists R.C \in \mathcal{A}$; $\nexists y:\{xRy, y:C\} \subseteq \mathcal{A}$	y – descendant x; $\mathcal{A}' = \mathcal{A} \cup \{xRy, y:C\}$
$\forall$-rule	point x is active; $x:\forall R.C \in \mathcal{A}$; $\exists y: xRy \in \mathcal{A} \wedge y:C \notin \mathcal{A}$	$\mathcal{A}' = \mathcal{A} \cup \{y:C\}$
T-rule	point x is active; $x:E \notin \mathcal{A}$, where $\top \subseteq E \in T$	$\mathcal{A}' = \mathcal{A} \cup \{x:E\}$

Given a concept C_0 this algorithm check if this concept is executable within existing terminology T, i.e. if there’s any interpretation where $C_0^{\mathcal{I}} \neq \varnothing$.

From the original ABox $\mathcal{A}_0 = \{x : C_0\}$, by applying the rules described above, one builds a search tree with a $\mathcal{A}_0$ root and 0, 1, or 2 descendants for each node (ABox). Application of the rules is terminated if none of the rules can be applied to the next ABox $\mathcal{A}$, or if there is a clear contradiction in $\mathcal{A}$ (i.e. for some individual x and concept C $\{x : C, x : \neg C\} \subseteq \mathcal{A}$ or $\{x : \bot\} \subseteq \mathcal{A}$).

To fulfill the terminality condition, the idea of an active point is introduced for $\exists$-rule and T-rule together not to lead to the infinite generation of individuals with the same set of concepts of their belonging.

Definition 7. A point x blocks the point y if it is an ancestor of y and $L(x) \supseteq L(y)$ where $L(x) = \{C | x : C \in \mathcal{A}\}$. The point y is called blocked if it is blocked by any point x. A point is called active if it is neither blocked nor descended of any blocked point.

Tableau algorithm for $\mathcal{ALCQ}$

The above described tableau algorithm for $\mathcal{ALC}$ logic can be extended for the case of $\mathcal{ALCQ}$ logic. Here we suppose that signs <, >, = are expressed based on $\leq$, $\geq$ signs as it was mentioned before in the definition of $\mathcal{ALCQ}$ logic. Updated set of rules for tableau algorithm is presented in Table 2.

Table 2. Rules of $\mathcal{ALCQ}$+TBox tableau algorithm

Rule	Conditions of applying	Action
$\sqcap$-rule	point x is active; $x : (C \sqcap D) \in \mathcal{A}$	$\mathcal{A}' = \mathcal{A} \cup \{x : C, x : D\}$
$\sqcup$-rule	point x is active; $x : (C \sqcup D) \in \mathcal{A}$	$\mathcal{A}' = \mathcal{A} \cup \{x : C\}$, $\mathcal{A}'' = \mathcal{A} \cup \{x : D\}$

Rule	Conditions of applying	Action
$\forall$-rule	point x is active; $x:\forall R.C \in \mathcal{A}$; $\exists y: xRy \in \mathcal{A} \wedge y:C \notin \mathcal{A}$	$\mathcal{A}' = \mathcal{A} \cup \{y:C\}$
T-rule	point x is active; $x:E \notin \mathcal{A}$, where $\mathcal{T} \subseteq E \in T$	$\mathcal{A}' = \mathcal{A} \cup \{x:E\}$
$\geq$-rule	point x is active; $x:\geq nR.C \in \mathcal{A}$; $\lvert\{d:\{d:C, xRd\} \subseteq \mathcal{A}\}\rvert = m < n$	$y_{m+1},...,y_n$ – descendants of x; $\mathcal{A}' = \mathcal{A} \cup \left\{\begin{matrix} xRy_{m+1}, y_{m+1}:C, \\ ..., \\ xRy_n, y_n:C \end{matrix}\right\}$ $\mathbb{C}' = \mathbb{C} \cup y_{m+1} \neq ... \neq y_n$
$\leq$-rule	point x is active; $x:\leq nR.C \in \mathcal{A}$; $\{x_1:C, xRx_1, ..., x_m:C, xRx_m\} \subseteq \mathcal{A}$; $m > n$	for each pair x_i, x_j that $x_i \neq x_j \notin \mathbb{C} : \mathcal{A}^{ij} = \mathcal{A} \vert x_j \to x_i$, where $\mathcal{A} \vert x_j \to x_i$ is ABox where all occurrences of x_j are substituted by x_i if there are no such pairs – terminate the algorithm with contradiction

Here $\geq$-rule absorbs $\exists$-rule defined above, as far as $\exists R.C = (\geq 1 R.C)$.

Here we introduce an additional set $\mathbb{C}$ of individuals constraints of a form $x_i \neq x_j$ that note which individuals must not be equal. This set is used in $\leq$-rule to join individuals in order to comply with 'no more than' constraint defined by this rule.

Application of the rules is terminated if none of the rules can be applied to the next ABox $\mathcal{A}$, or if there is a clear contradiction in $\mathcal{A}$, or as a result of a contradiction inside $\leq$-rule.

Semantic contradictions determination

We will say that a constructed knowledge base has a *contradiction* if any of the atomic concepts is not executable within its terminology. Contradictions are quite likely to appear when some errors happened during syntax and/or semantic parsing of a natural language text. Yes, from the extraction process given above we can define that in any non-error case each named concept should have at least one individual that represents it.

Let's consider a knowledge base constructed above. For efficient executability check we first find out which concept names are leaves in inclusion tree of the knowledge base: that is which concept names are not in the right side of any of inclusions. We can construct a graph based on terminology axioms to determine leaves like in Figure 13.

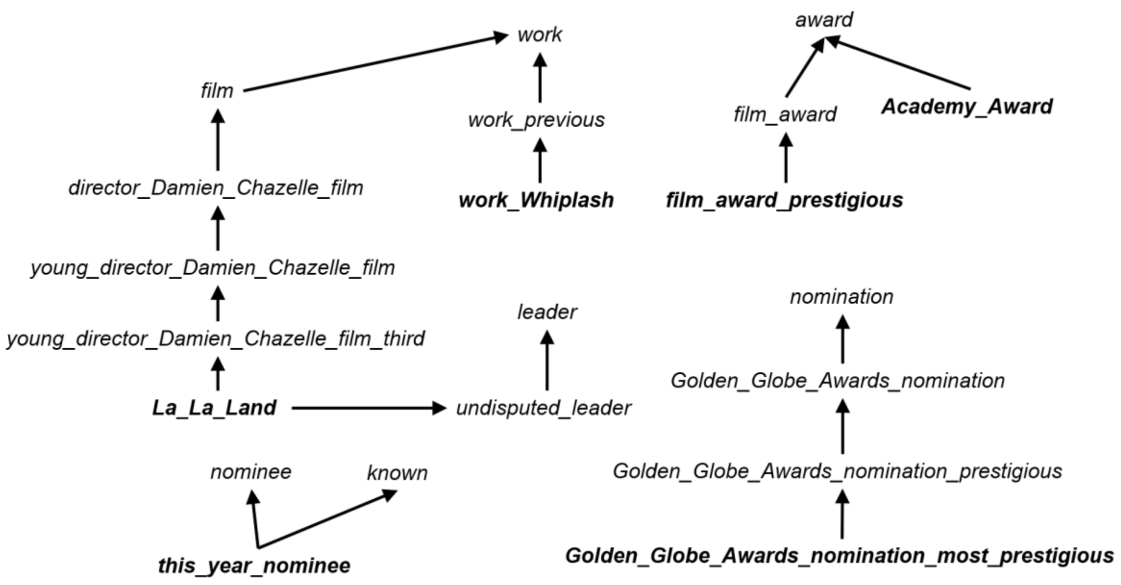

work
award
film
work_previous
film_award
Academy_Award
director_Damien_Chazelle_film
work_Whiplash
film_award_prestigious
young_director_Damien_Chazelle_film
nomination
leader
young_director_Damien_Chazelle_film_third
Golden_Globe_Awards_nomination
La_La_Land
undisputed_leader
nominee
known
Golden_Globe_Awards_nomination_prestigious
this_year_nominee
Golden_Globe_Awards_nomination_most_prestigious

Figure 13. Concept inclusion graph

For our knowledge base we will get a following set:

{*work_Whiplash*, *film_award_prestigious*, *Academy_Award*,
La_La_Land, this_year_nominee}

For each of the 'leave' concept names we add associated individual and fact to the original ABox as follows:

$$\begin{aligned}&\mathcal{A}_0 = \{a_1 : work\_Whiplash, a_2 : film\_award\_prestigious, a_3 : Academy\_Award,\\ &a_4 : La\_La\_Land,\ a_5 : this\_year\_nominee,\\ &a_6 : Golden\_Globe\_Awards\_nomination\_most\_prestigious\}\end{aligned}$$

And then we populate our ABox with all facts coming up from all inclusion and equivalency axioms. From this stage we have constructed an ABox with all named concepts presented by exactly one individual. So, if the tableau algorithm will give a contradiction – some named concept is non-executable, and if not – all of them are executable and knowledge base is consistent.

$$\begin{aligned}&\mathcal{A}_0 = \{a_1 : work\_Whiplash, a_1 : work\_previous, a_1 : work,\\ &a_1 :\geq 2R_{win}.film\_award\_prestigious, a_1 := 3R_{win}.Academy\_Award,\\ &a_2 : film\_award\_prestigious, a_2 : film\_award, a_2 : award,\\ &a_3 : Academy\_Award, a_3 : Academy\_Award,\\ &a_4 : La\_La\_Land, a_4 : young\_director\_Damien\_Chazelle\_film\_third,\\ &a_4 : young\_director\_Damien\_Chazelle\_film, a_4 : director\_Damien\_Chazelle\_film,\\ &a_4 : undisputed\_leader, a_4 : leader, a_4 : leader, a_4 := 14R_{has}.nomination,\\ &a_4 : film, a_4 : work, a_5 : this\_year\_nominee, a_5 : nominee, a_5 : known,\\ &a_6 : Golden\_Globe\_Awards\_nomination\_most\_prestigious,\\ &a_6 : Golden\_Globe\_Awards\_nomination\_prestigious,\\ &a_6 : Golden\_Globe\_Awards\_nomination, a_6 : nomination,\\ &a_4 : \exists R_{win}.Golden\_Globe\_Awards\_nomination\_most\_prestigious\}\end{aligned}$$

Let’s proceed now a tableau algorithm starting from original ABox $\mathcal{A}_0$.

Consider fact $a_1 : \geq 2R_{win}.film\_award\_prestigious$. After applying of $\geq$-rule to this fact, ABox will be updated as follows:

$$\begin{aligned}&\mathcal{A}_1 = \mathcal{A}_0 \cup \{a_7 : film\_award\_prestigious, a_1 R_{win} a_7,\\ &a_8 : film\_award\_prestigious, a_1 R_{win} a_8, a_7 \neq a_8\}\end{aligned}$$

Consider fact $a_1 : \geq 3R_{win}.Academy\_Award$. After applying of $\geq$-rule to this fact, ABox will be updated as follows:

$$\begin{aligned}&\mathcal{A}_2 = \mathcal{A}_1 \cup \{a_9 : Academy\_Award, a_1 R_{win} a_9,\\ &a_{10} : Academy\_Award, a_1 R_{win} a_{10},\\ &a_{11} : Academy\_Award, a_1 R_{win} a_{11}, a_9 \neq a_{10} \neq a_{11}\}\end{aligned}$$

Consider fact $a_4 : \geq 14R_{has}.nomination$. After applying of $\geq$-rule to this fact, ABox will be updated as follows:

$$\begin{aligned}&\mathcal{A}_3 = \mathcal{A}_2 \cup \{a_{12} : film\_award\_prestigious, a_4 R_{has} a_{12}, ...,\\ &a_{25} : film\_award\_prestigious, a_4 R_{has} a_{25}, a_{12} \neq ... \neq a_{25}\}\end{aligned}$$

Consider fact $a_4 : \geq 1R_{win}.Golden\_Globe\_Awards\_nomination\_most\_prestigious$. After applying of $\geq$-rule to this fact, ABox will be updated as follows:

$$\mathcal{A}_4 = \mathcal{A}_3 \cup \{a_{26} : Golden\_Globe\_Awards\_nomination\_most\_prestigious, a_4 R_{win} a_{26}\}$$

Now, while no rules can be applied, let’s check $\leq$-conditions. It is clear that both conditions for facts $a_1 : \leq 3R_{win}.Academy\_Award$ and $a_4 : \leq 14R_{has}.nomination$ are satisfied.

Tableau algorithm for each of atomic concepts extracted from the text discussed above does not lead to any contradictions, which means its consistency. We must notice that we can't talk about accurate correctness here as far as we operate only with knowledge defined in the text, but not of the whole field it represents.

Let's now add axiom $work\_Whiplash \sqcap \exists R_{win}.Academy\_Award \sqsubseteq \perp$, which means that Whiplash film has won none of Academy Awards. Let's check if *work_Whiplash* concept is executable within this new terminology.

Algorithm starts from $\mathcal{A}_0 = \{x : work\_Whiplash\}$. After applying all terminology axioms we will get ABox $\mathcal{A}_0 = \{x : work\_Whiplash, x : \exists R_{win}.Academy\_Award\}$ and then $\mathcal{A}_0 = \{x : \perp\}$. We got a clear contradiction, so the concept in non-executable and the corresponding knowledge base is inconsistent.

Question answering

To formulate queries, we will introduce a new variety of characters – a finite set of individual variables $Var = \{x_0, x_1, \ldots\}$. We call query of a kind $u : C$ or uRv an atomic, where C is a concept, R – role, $u, v \in IN \cup Var$.

Definition 8. A conjunctive query is an expression of the form $\exists \bar{v}(t_1 \wedge \ldots \wedge t_k)$*, where* t_i *are atoms,* $\bar{v} = \{v_1, \ldots, v_l\}$ *is a list of some of the variables included in* t_i*. Variables* v_i *are called bound and the rest of the variables are called free. If* $\bar{v} = \{v_1, \ldots, v_l\}$ *is a list of free variables of the query* q*, we will write it as* $q(\bar{v})$*.*

Consider the above constructed knowledge base. Natural language query *Which films won the Academy Award?* can be written as follows: $q(x) = x : \exists R_{win}.$ *Academy_Award*. The answer to the query is the set of individuals, which satisfy the query conditions, along with a set of concepts they belong to. For the above example, the answer is $\{a_1 : work\_Whiplash, a_2 : La\_La\_Land\}$. We can interpret this answer into natural language response: *Work 'Whiplash' and La La Land*.

It should be noted that the theory of knowledge bases is based on the belief in the openness of the world: the knowledge base is a collection of all models in which the axioms given in it are valid. Therefore, the query answer is always a subset of an accurate answer to a natural-language question, as opposed to a query to a database, which is always an exactly complete answer to a question. [Kryvyi et al, 2018]

Conclusion

The existing state of the natural language processing field provides qualitative input for the task of natural language knowledge base population. Yes, the dependency tree, built according to the Universal Dependencies framework, allows to extract the terminological axioms and facts of the knowledge base, including those with numerical constraints. However, the unresolved problem of finding correspondences for the Ukrainian language does not allow us to speak about a sufficiently high-quality state of solving the knowledge bases population problem with Ukrainian-language texts, which confirms the need to work on the correspondence corpus for the Ukrainian language.

In this article we reviewed current state of modern natural language processing tasks that can provide qualitative input for knowledge base population. We provided the simple approach to populate a knowledge base from Universal Dependencies and coreferences trees, that will be unified and improved in future works. On a constructed base we showed how tableau algorithm can be used for consistency checking and finding semantical errors made during text analysis. Also, some advices for question answering based on description logics are given, too.

The above described approach to the population of the knowledge bases can be extended to cases of conditional sentences and causations and adapted to the different temporal context of the statements made in the text. Accordingly, the analysis of knowledge bases containing such information requires the use of an extended apparatus of description logics, including their combination with temporal logics and the use of an additional kinds of terminological axioms.

Bibliography

[Baader et al, 2007] Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P. The Description Logic Handbook. – Cambridge University Press. – 2007. – P. 578.

[Bohnet et al, 2018] Bohnet B., McDonald R., Simões G., Andor D., Pitler E., Maynez J. Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings. – Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. – 2018. – P. 2642-2652.

[Cetto et al, 2018] Cetto M., Niklaus C., Freitas A., Handschuh S. Graphene: A Context-Preserving Open Information Extraction System. – Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations. – Santa Fe, New Mexico. – 2018. – P. 94–98.

[Darchuk, 2013] Darchuk N. Automated syntax analysis of texts from Ukrainian language corpus. – Ukrainian linguistics. – 2013. - № 43. – P. 11-19. (In Ukrainian)

[Devlin et al, 2019] Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. – NAACL-HLT. – 2019. – P. 4171–4186.

[Kryvyi et al, 2018] Kryvyi S., Darchuk N., Provotar O. Onlogoly-based systems of natural language analysis. – Problems of programming. – 2018. – № 2-3. – P. 132-139. (In Ukrainian)

[Léchelle et al, 2019] Léchelle W., Gotti F., Langlais P. WiRe57: A Fine-Grained Benchmark for Open Information Extraction. – 2019. – P. 6–15.

[Manning et al, 2002] Manning C., Grow T., Grenager T., Finkel J., Bauer J. Stanford Tokenizer. – 2002.

[McDonald et al, 2003] McDonald R., Nivre J., Quirmbach-Brundage Y., Goldberg Y., Das D., Ganchev K., Hall K., Petrov S., Zhang H., Täckström

O., Bedini C., Castelló N. B., Lee J. Universal Dependency Annotation for Multilingual Parsing. – In Proceedings of ACL. – 2003. – P. 92–97.

[Mrini et al, 2019] Mrini K., Dernoncourt F., Bui T., Chang W., Nakashole N. Rethinking Self-Attention: An Interpretable Self-Attentive Encoder-Decoder Parser. – 2019.

[Niklaus et al, 2002] Niklaus C., Cetto M., Freitas A., Handschuh S. A Survey on Open Information Extraction. – Proceedings of the 27th International Conference on Computational Linguistics. – Santa Fe, New Mexico, USA. – 2018. – P. 3866–3878.

[Palagin et al, 2012] Palagin O., Kryvyi S., Petrenko N. On the automation of the process of extracting knowledge from natural language texts. – Natural and Artificial Intelligence Intern. Book Series. – Inteligent Processing. – ITHEA. – Sofia. – N 9. – 2012. – P. 44–52. (In Russian)

[Stanovsky and Dagan, 2016] Stanovsky G., Dagan I. Creating a Large Benchmark for Open Information Extraction. – Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. – Austin, Texas, US. – 2016. – P. 2300–2305.

[Zhan and Zhao, 2019] Zhan J., Zhao H. Span Based Open Information Extraction. – 2019.

Authors' Information

Serhii Kryvyi *– Faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv, Ukraine; Doctor of Physical and Mathematical Sciences, Professor; e-mail: sl.krivoi@gmail.com*

Major Fields of Scientific Research: automata theory, theory of Petri nets, software verification, discrete mathematics, computational complexity theory, recursion theory, technology of software creation, applied mathematical logic, graph theory, cryptography.

Nataliia Darchuk *– Institute of Philology, Taras Shevchenko National University of Kyiv, Ukraine; Doctor of Philological Sciences, Professor; e-mail: nataliadarchuk@gmail.com*

Major Fields of Scientific Research: computer linguistics, computer lexicography, corpus linguistics, linguistics, theoretical grammar.

Hryhorii Hoherchak *– Faculty of Computer Science and Cybernetics, Taras Shevchenko National University of Kyiv, Ukraine; PhD student; e-mail: gogerchak.g@gmail.com*

Major Fields of Scientific Research: natural language processing, knowledge base and database systems, applied logics.