UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Reio Laabus

Extracting semantic propositions from dependency trees

Bachelor's Thesis (9 ECTS)

Supervisor: Kairit Sirts

Tartu 2017

Extracting semantic propositions from dependency trees

Abstract:

The main goal of this thesis is to implement a tool for extracting propositions from dependency parse trees. Propositions are part of sentences that describe the ideas what people want to express. Finding the propositions and counting them has been found to be good measurement to relate it with readability, memory or prediction of Alzheimer's disease. Earlier works have extracted the propositions manually, my program called Proposition Count based on Patterns, short for PCP, does it automatically using patterns. Patterns are regular expressions that are created on the basis of AID manual and they are classified into 3 groups: predications, modifications and connectives. Patterns are used on dependency parse trees that present the syntactic structure of a sentence. It has been found that the dependency structure and propositions suit more naturally and is direct. The results depend a lot on correctness of the sentence, because parsers are not able to correctly parse faulty sentence and patterns can't extract correct propositions from incorrect sentences. Results are also affected by what parser is being used, if using different parser than I used with the same patterns, then the possibility that extracting different count of propositions is high.

Keywords:

Information extraction, semantic proposition, dependency parsing

CERCS: P175 Informatics, systems theory

Sõltuvussüntaksi puudest semantiliste propositsioonide leidmine

Lühikokkuvõte:

Käesoleva bakalaureuse töö eesmärk on luua teek, mis loendab ja väljastab sõltuvussüntaksi puudelt mustrite abil propositsioonid. Propositsioonid on lause osad, mis kirjeldavad ideid, mida antud lausega tahetakse edasi anda. On leitud, et propositsioonide leidmine ja loendamine on heaks mõõduks, et seostada propositsioonide arvu loetavuse, mälu või Alzheimeri haiguse ennustamisega. Varasemalt on propositsioone lausetest leitud manuaalselt, minu programm PCP teeb seda automaatselt mustreid kasutades. Mustrid on regulaaravaldised, mis on vastavalt AID manuaalile koostatud ja nad jaotatakse kolme suuremasse gruppi: predikatsioonid, modifikatsioonid ja ühendajad. Mustreid kasutatakse sõltuvussüntaksi puudel, mis esitavad lause süntaktilist struktuuri. Sõltuvusstruktuuri ja propositsiooide struktuuri ehitus on omavahel sarnane. Kuna parserid ei oska vigast lauset parsida, mille tõttu ka mustrid ei leia õigeid propositsioone, siis tulemused sõltuvad lause ehituse korrektsusest. Samuti, kuna erinevad parserid töötlevad lauseid erinevalt, siis on suur tõenäosus, et ka propositsioonide arv võib erineda.

Võtmesõnad:

Informatsiooni ekstraheerimine, semantiline propositsioon, sõltuvussüntaksi analüüs

CERCS: P175 Informaatika, süsteemiteooria

Contents

1. Introduction

Humans use natural language to give, inform and acquire information about their thoughts, feelings or wishes that is to convey ideas, which are called semantic propositions. To do this, language uses symbols to form words and words to form sentences. Symbols consists of two parts, a form and a meaning. So by speaking or writing we want to transfer the meanings of that text or speech, one way of quantifying the meaning, introduced and researched by Kintsch [7], is to count the number of propositions.

According to Kintsch [7] a proposition or an idea is the basic unit of memory for text. Propositions can be grouped into three major classes: predications, which describe actions with actor and object, modifications, which modify an item or an event and connectives, which connect two propositions.

Idea density (ID) of a text or speech is computed as the number of expressed propositions divided by the number of words. In some cases ID is also calculated for each 10 words of text [9]. It shows the amount of information that is conveyed relative to the number of words used. So, it expresses numerically how much information a sentence provides and the efficiency of it. In sentences with the same length the higher ID means this sentence conveys more information, while lower ID implies that the sentence may contain repetition and the meaning may be senseless [2]. Experiments have been conducted that relate ID to readability [7] and memory [8], which was the main interest of Kintsch, to develop a quantitative measure of memory strain when memorizing a text. ID has been also related to aging [9] and has been most researched in association with Alzheimer's disease [3, 10, 11].

The most cited research related to ID is the so-called nun-study [3, 10, 11], where for a period of 11 years 678 nuns had annual cognitive, behavioural and neurological examinations. After death their brains were analyzed for shrinkage of cerebral cortex, shrinkage of hippocampus and other tangles in the brain, which are used to diagnose Alzheimer's disease. Then ID was used to analyze the participants' early life written narratives and an association was found between the low ID value and the development of Alzheimer's disease at a later age.

Traditionally most psycholinguistic researches have manually counted propositions. Since counting manually is time consuming and troublesome CPIDR was developed, what has been used in many studies. CPIDR, what is described more specifically in section 3.1 doesn't extract the propositions, it only counts them. Recently has been developed IDD3[1], what is described in section 3.2, which as in this work extracts propositions on the basis of AID manual [1]. Both works use rule based systems, but I create rules as patterns what can be used on patterns to find propositions.

The goal of this project is to implement a tool that can extract the propositions from the natural language text based on dependency parse trees. Dependency parse trees express the syntactic structure of sentences using directed graphs. Earlier works [1] have noticed that there are similarities between dependency parse trees and semantic propositions. To extract propositions from dependency parse tree I use Semgrex [12] that is part of Stanford Core NLP library [13]. Semgrex is a utility for matching patterns in dependency graphs. Patterns that are written in Java using regular expressions are the main elements of this thesis and they are constructed on the basis of rules written in Chand et al.'s AID manual [2].

This thesis is divided into five chapters, the first part describes two important definitions that are used frequently throughout this study: proposition and dependency parse tree. The

[1] Code for IDD3 is available but not functional

second part is about earlier works CPIDR and IDD3 that have been created and their difference in implementation and finding propositions. The third part describes what most frequently used patterns are and how they are constructed. Patterns are the units used to get propositions out of sentences. The fourth part describes the implementation, what other tools are used and how propositions are found. The last part is evaluation, where I compare results with manually counted propositions and also with earlier works. I also discuss what made it difficult to find propositions using patterns and dependency parse trees and what can be made to get better results.

2. Review of definitions and terms

In this section I first describe more specifically what propositions are, how they are classified. Then I will give a deeper overview on dependency parse tree, what it is and how it looks like.

2.1 Proposition

Propositions are word tuples that describe the ideas expressed in a sentence. Sentences can consist of several propositions, each representing a single idea. According to Kintsch and Keenan [7] propositions are those parts of the text that show how people remember and understand texts.

Table 1. Propositions

Sentence:	
I felt sick because I ate rotten food.	
Propositions:	
1.	felt, I, sick
2.	ate, I, food
3.	food, rotten
4.	because, 1, 2

For example in table 1 for the sentence "I felt sick because I ate rotten food" there are extracted 4 propositions: "felt, I, sick", "ate, I, food", "food, rotten" and "because, 1, 2". Numbers in last proposition represent what propositions are connected by the connective word "because".

Semantic propositions differ from logical propositions at least two ways. First, the verbss modal, tense or aspect markers are not counted as separate propositions. Second, common nouns are not counted as propositions. That are words that refer to a person, place or thing but is not the name of it.

Turner and Greene [5] developed further the ideas provided by Kintsch and made a manual for extracting propositions. They divided propositions into three classes: predications, modifications and connectives. Predication propositions express ideas of actions or states, they consist of a predicator followed by its arguments, for instance in table 1 first proposition belongs under predication. Modification propositions modify an item or event by restricting it, quantifying it or limiting it, they consist of an attribute that modifies an entity, in table 1 third proposition belongs to modification class. Connective propositions connect propositions in the text to each other, they consists of a word or phrase that link these propositions, for instance in table 1 fourth proposition is of this type and it connects propositions 1 and 2.

2.2 Dependency parse tree

Parse trees are used in natural language processing to present the syntactic structure of a sentence.

Dependency parsing is one way of representing the syntactic structure of a sentence. A dependency parse tree is a directed graph where vertices are words and arcs correspond to syntactic dependency relations between these words. All dependency parse trees have one root node or parent node which is usually the main verb of the sentence which does not have a governor. Each child node must have exactly one parent node but each node can have zero or more dependent nodes. Figure 1 shows the dependency tree for an example sentence "I felt sick because I ate rotten food". The root of the sentence is "felt", it doesn't have any

governors, thus no arrows point to that word. Words “I”, “sick” and “ate” are direct children of the root word, they are dependent of the root word “felt” with syntactic relations which are written on the vertexes. For example the relation “nsubj” means that the word “I” is the nominal subject for the verb “ate”.

There are about 40 different relation types, the exact number differs depending on which dependency system is used.

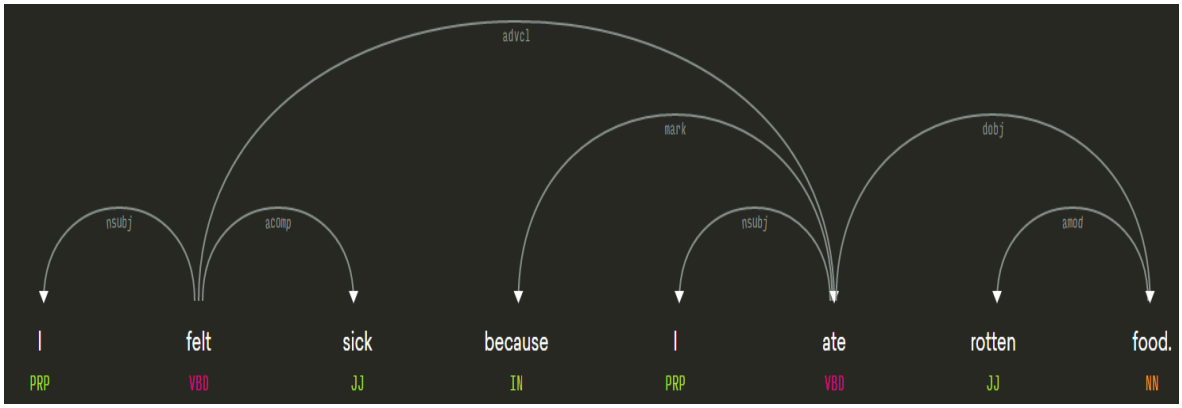

Figure 1. Dependency parse tree[2]

[2] All dependency parse tree images are from https://demos.explosion.ai/displacy/

3. Earlier works

As of now there are two known programs related to automatically discovering propositions from English sentences. The Computerized Propositional Idea Density Rater short for CPIDR (pronounced „spider“) [4] and Propositional Idea Density from dependency Trees short for IDD3 [1].

3.1 CPIDR

CPIDR [4] counts propositions from text by using part of speech (POS) tagging. For tagging CPIDR uses MontyLingua tagger [15], on tagged sentences CPIDR uses predefined rules to count propositions. This system aimed at replicating the proposition counts described by Turner and Greene [5]. CPIDR defines its rules based on Snowden et al. [3] where propositions conform to certain parts of speech and each proposition is usually either a verb, adjective, adverb prepositional verb or connection between sentences. Additionally CPIDR 3 uses a number of readjustment rules to obtain more precise outcome. For example “to” +VERB is counted as one proposition, modals are not counted as separate propositions unless they are negated, copula is counted as separate proposition when it introduces a noun phrase. CPIDR also rearranges subject-auxiliary inversion to handle questions.

In addition to validation on Turner and Greene examples [5] it was also tested against human raters, using samples from volunteers. In those sentences non-lexical fillers, utterances that repeated the interviewer were eliminated and 10 last sentences of each speech were selected for analysis.

The main difference between CPIDR and my solution is that CPIDR uses part-of-speech tags to count the propositions while I use dependency structure. Also in PCP I print out the propositions that were found.

3.2 IDD3

IDD3 [1] counts and extracts propositions using dependency parse trees, for that it uses rules and rule sets. Each rule set has access to variables like word, POS, head, children and functions like process, emit and return. IDD3 uses Stanford parser [11] to get dependency parse trees from sentences. Rules were created according to Chand et al.’s manual [2]. IDD3 was evaluated on four texts, for which the propositions were manually counted. According to the authors of IDD3, IDD3 extracts propositions very accurately from well-formed texts.

The difference between PCP and IDD3 mainly relays in how the propositions are extracted. In PCP I use patterns that search for match, also parsers that construct the dependency parse trees are different, in PCP spaCy [14] parser is used.

4. Patterns

Constructing patterns to find propositions from sentences is the main work of this thesis. The developed patterns are based on the AID manual [2], which clarifies and improves the rules written by Turner and Greene [5]. Similar to Turner and Greene the patterns are divided into three subtypes: predication, modification and connectives. Predication ideas express an action or state, it includes the subject who is doing something and optional object on what something is done, for example "I ate food". Modification ideas provide more information about an item or event, they can specify ("blue sky"), quantify ("one child"), modify an action ("ate slowly") or negate the idea ("he can't sleep"). Connective ideas relate two or more propositions or show relation between them ("the child fell and is crying"). All the patterns I used to identify propositions are in appendix A.

I wrote total of 46 different patterns, in which 10 belong under predication, 11 are of type conjunction and 25 patterns find propositions that are related to modification. Further information about what patterns are more frequently used and which are only for some special cases can be seen in paragraph 6.2.

4.1 Predicative patterns

4.1.1 Pred1

Pattern Pred1 is the main predicative pattern that extracts a verb with its subject and direct object, where the object can be omitted. For example in the sentence „I ate food" shown in figure 2, the word „ate" is action, the words „I" and „food" are the subject and the object respectively. The verbs' modal, tense or aspect markers are not counted as separate ideas, both „I slept today" and „I had slept today" constitute of one proposition. The same applies to auxiliary (AUX) for example in the sentence "He has been there", to passive auxiliary (AUXPASS) and to phrasal verb particle (PRT) for example in the sentence "I grew up in Korea", which are counted under the main proposition. Determiner relation (DET), that is dependent on subjects or objects are also written under one idea like in the example "The cat is big", same applies to possession modifiers (POSS) for example in the sentence "His car is red" on subjects and compound on object predicates (OPRD).

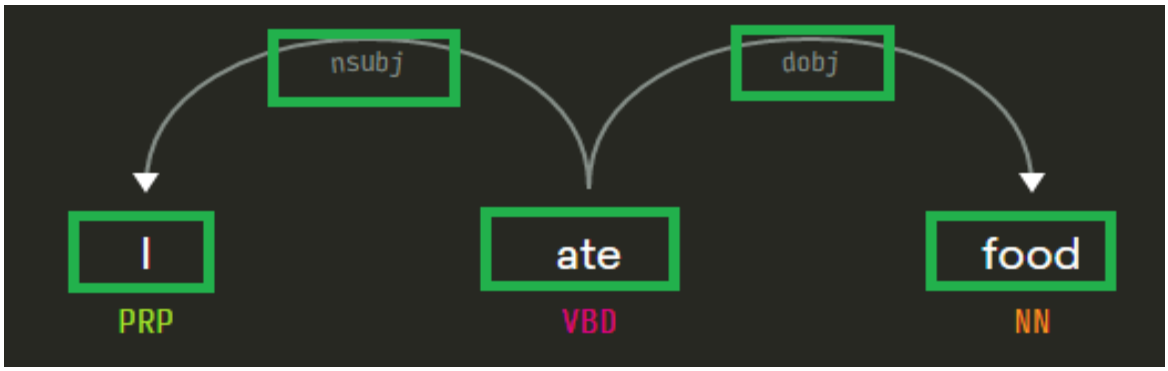

Figure 2. Root verb with dependencies NSUBJ and DOBJ

In Pred1, instead of direct object (DOBJ) or object predicate (OPRD) it can be replaced with adjectival complement (ACOMP) for example in the sentence "The dog is funny" or adverbial modifier (ADVMOD), which can have optional dependant noun phrase as adverbial modifier (NPADVMOD) for example in the sentence "She is 5 years old". Also attribute with its optional dependant numeric modifier (NUMMOD) and interjection (INTJ) can replace the object and its dependants.

Because gerunds take multiple forms, they can act as subjects („running is good for me“), as it is in figure 3, adjectives („The purring cat was on the bed“) as for example in figure 4 and objects („I like running“) or part of progressive verb form („He is snoring“).

Figure 3. Gerund as subject

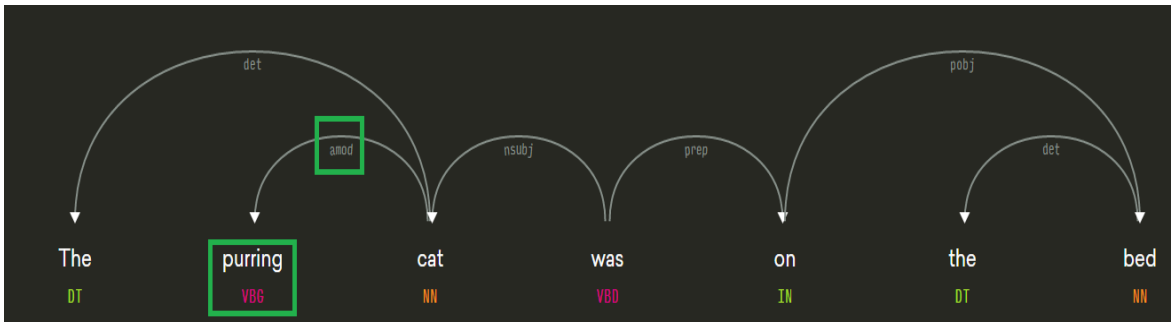

Figure 4. Gerund as adjectival modifier (AMOD)

If the gerund is part of a progressive construction it is not counted as a separate idea („She is jogging“). If it is not part of the main verb construction, then it is considered if gerundium acts as a noun or a verb. If it acts as a noun it is written as one idea („Cooking takes time“), except when the gerundium that is object of a preposition („We came from jogging“), in which case it is written as separate idea.

When gerund acts as a verb and is accompanied by its own object („Cooking chicken takes time“), like is the case in figure 5, then the object of the gerund is treated as a separate idea together with the gerund, which is obtained with pattern p6.

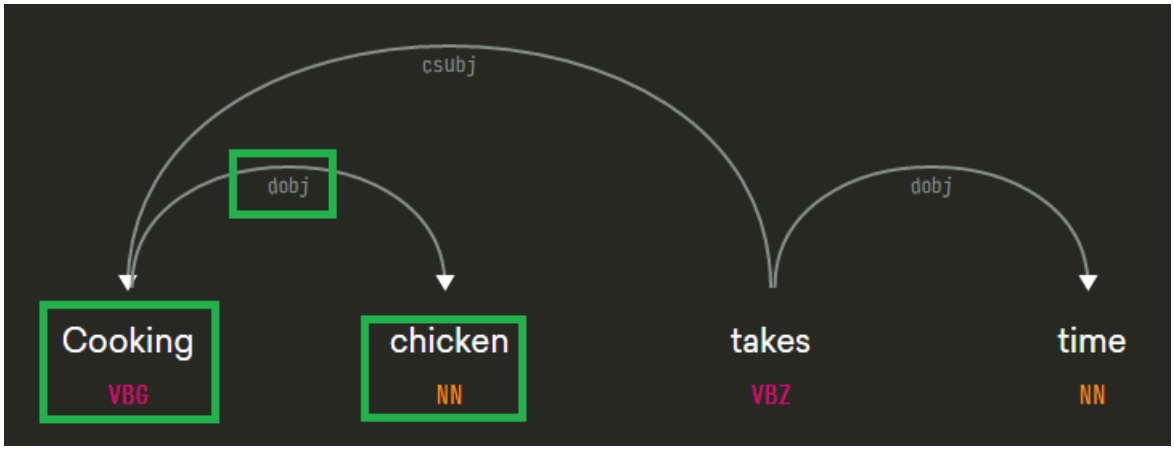

Figure 5. Gerund acting as verb and has direct object dependency

Words „some“ and „which“ that act like subjects in a sentence and have Wh-determiner (WTD) for example in the sentence „The cat that likes people“ or determiner (DT) as their POS tag are not extraced in the pattern Pred1. If a word has dependency of a direct object

relation and Wh-pronoun (WP) POS tag then this kind of structure is not counted as idea for pattern Pred1.

4.1.2 Pred2

Second most used predications pattern is pattern Pred2 that finds propositions from sentences that consist of some word followed by preposition which in turn is governor for object of a preposition or direct object relation. For example in the sentence „We came from jogging“, this pattern finds proposition „from jogging“ where „from“ is a prepositional modifier (PREP) and „jogging“ is an object (POBJ) as can be seen in figure 6.

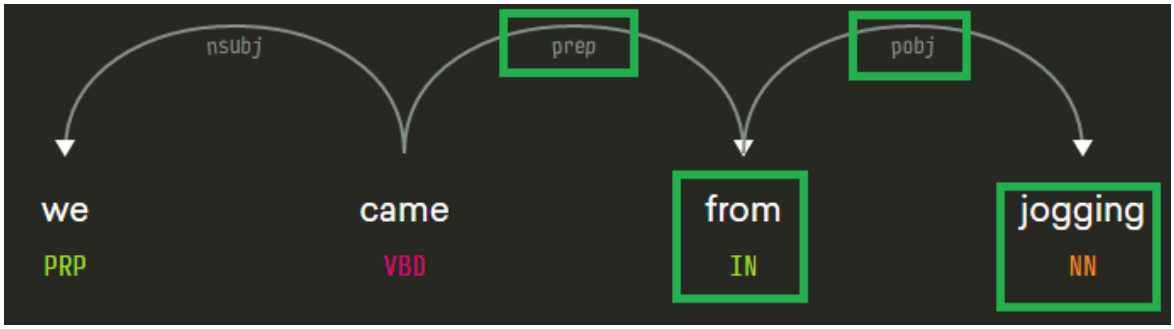

Figure 6. Verb has dependency PREP which has dependency POBJ

The object relation can have an optional determiner relation, which can only include words „the“, “a“ and “an“. It can also have compound (COMPOUND), adjectival modifier (AMOD), and conjunct (CONJ), which may not be noun (NN), dependent relation. Object relation can have two kinds of possession modifier relations, first a word that has noun (NN) POS tag and is governor of a determiner (DET) relation or a word that has personal pronoun (PRP) POS tag.

4.2 Modification patterns

4.2.1 Mod1

Pattern Mod1 finds ideas that belong to the modification class, they provide further detail regarding the specification of an item or event. This pattern finds propositions that consist of a word that is followed by nominal subject (NSUBJ) or direct object (DOBJ) which has adjectival modifier(AMOD) or possession modifier(POSS) as dependencies, also it can have optional case marking(CASE) relation. For example in the sentence „The fluffy cat was purring,“ this pattern finds proposition „cat fluffy“, where „cat“ is nominal subject(NSUBJ) and „fluffy“ is adjectival modifier(AMOD) like in figure 7.

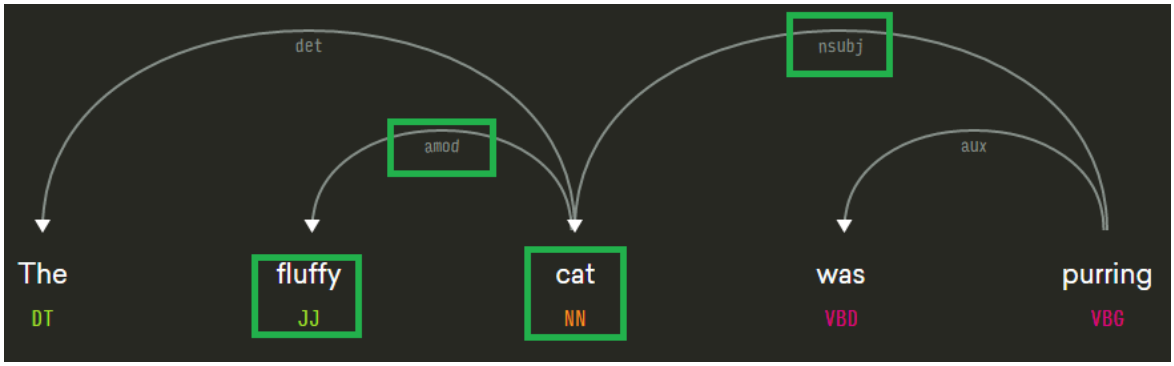

Figure 7. Verb has dependency NSUBJ which in turn has dependency AMOD

The word that has adjectival or possession modifier relation can only be some type of noun (NN) or adjective (JJ).

4.2.2 Mod2

Pattern Mod2 finds propositions from a sentence that consist of word that cannot be cardinal number (CD), adjective (JJ) or proper noun (NNP) followed by a word that is dependent of adverbial modifier(ADVMOD) relation. Adverbial modifier can also have a dependent word with optional negation modifier (NEG) relation and it may not have noun phrase as adverbial (NPADVMOD) modifier or prepositional (PREP) modifier. For example in the sentence „I worked there for 3 years“ this pattern finds proposition „there“ that is adverbial modifier, which can be seen in figure 8.

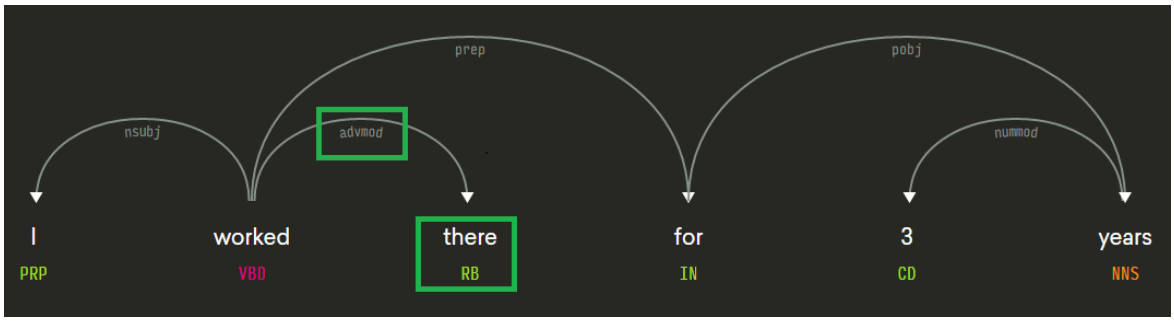

Figure 8. Verb with ADVMOD dependency

4.3 Conjunction patterns

4.3.1 Con1

Pattern Con1 belongs under the connectives category, it extracts propositions that connects two propositions that are related to each other by sequence where second action is caused by the first for example in the sentence “I cooked the pasta and served it” or by dependency, where first and second action are linked together by cause and effect, like in the example “The child fell and is crying”.

This pattern consists of a word that has to be root of the sentence, which is denoted by “$” sign, that word has to be followed by a word that is dependent through coordinating conjunction (CC) relation and conjunct (CONJ) relation, as in figure 9.

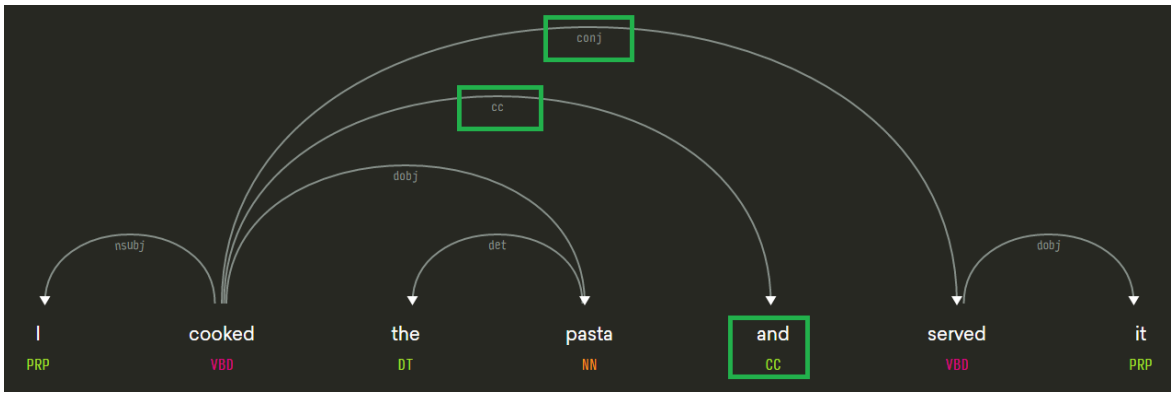

Figure 9. Root verb has dependencies CC and CONJ

Word that has conjunct relation with the root word, can’t be verb in 3rd singular present (VBZ) form or adjective (JJ) and it can have adverbial modifier relation (ADVMOD). This is to match the proposition count given in AID manual.

4.3.2 Con2

Pattern Con2 is connectives pattern that finds propositions where a word has adverbial clause modifier (ADVCL) relation that has a dependant with a marker (MARK) relation as it is in figure 10. For example in the sentence „She likes vanilla, whereas I like chocolate“ this pattern finds idea „whereas“ which connects two statements „she likes vanilla“ and „I like chocolate“.

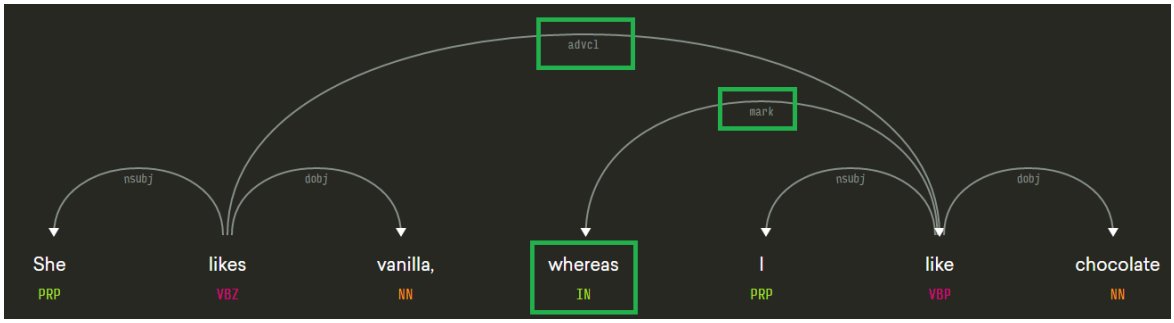

Figure 10. Word with ADVCL dependency what has MARK dependency

5. Implementation

To extract proposition from dependency parse trees I first need to parse sentences, for that I use spaCy[3] [14]. After that I use Semgrex [12] to construct patterns that search for a match from the parsed sentences.

5.1 Java – Finding propositions

Semgrex [12] is utility for matching patterns in dependency parse trees, it is part of Stanford Core NLP [13] toolkit. For finding propositions I first read in CoNLL formatted file and construct SemanticGraph object from each sentence. Then I use patterns to find match on SemanticGraph object. If match is found then that is one proposition, if there is no match, this means that in currently processed sentence there is no proposition with that structure and next pattern is used.

Patterns structure consist of braces ({}), which is any node, nodes can have additional attributes to specify what POS tag or word a node can be. Nodes have relationships between them, which have two parts: the relation symbol and the relation type. For example in structure "A<reln B" A is the governor of a *reln* relation with B. Nodes can be grouped with parentheses and named with equals ("=") sign, which can be used to retrieve the string form of the relation if found in the graph. To make pattern structures with optional relations to nodes question mark ("?") should be used.

5.2 Python – CoNLL format

To extract propositions from dependency parse trees, I first need to parse sentences, for this I used spaCy, what is a Python library for natural language processes. At first I also experimented with Stanford parser which is part of Stanford Core NLP [13], but I found that spaCy parse tree is more similar to the Chand et al. manual rules [2] for finding propositions. I used a small script which takes in a text file with sentences and puts out a text file where the sentences are written in CoNLL format.

CoNLL format is universal non-graphical way to represent dependency parse tree in tabular form. CoNLL format sentences consists of one or more word lines and word lines contain following fields:

1. ID: Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes.
2. FORM: Word form or punctuation symbol.
3. LEMMA: Lemma or stem of word form.
4. UPOSTAG: Universal part-of-speech tag.
5. XPOSTAG: Language-specific part-of-speech tag; underscore if not available.
6. FEATS: List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available.
7. HEAD: Head of the current word, which is either a value of ID or zero (0).
8. DEPREL: Universal dependency relation to the HEAD (root if HEAD = 0) or a defined language-specific subtype of one.
9. DEPS: Enhanced dependency graph in the form of a list of HEAD-DEPREL pairs.
10. MISC: Any other annotation.

Because Java Stanford Core NLP doesn't need all of the fields information to create dependency parse tree, Python program doesn't find FEATS, DEPS and MISC.

[3] https://spacy.io/

Table 2. CoNLL format for the sentence „I felt sick because I ate rotten food”

ID	FORM	LEMMA	UPOSTAG	XPOSTAG	FEATS	HEAD	DEPREL	DEPS	MISC
1	I	i	PRON	PRP	_	2	nsubj	_	_
2	Felt	feel	VERB	VBD	_	0	root	_	_
3	Sick	sick	ADJ	JJ	_	2	acomp	_	_
4	Be- cause	because	ADP	IN	_	6	mark	_	_
5	I	i	PRON	PRP	_	6	nsubj	_	_
6	Ate	eat	VERB	VBD	_	2	advcl	_	_
7	Rotten	rotten	ADJ	JJ	_	8	amod	_	_
8	Food	food	NOUN	NN	_	6	dobj	_	_
9	.	.	PUNCT	.	_	2	punct	_	_

6. Evaluation

To evaluate the programs performance, I used 177 sentences given by Chand et al.'s AID manual [2] and on 97 sentences given by Turner and Greene [5] to compare the number of propositions PCP counted against how many the manuals themselves counted. In the Chand et al.'s manual there are also utterances that are grammatically wrong, incomplete or have non-lexical fillers and repetitions, which are hard for PCP to analyse correctly because parser doesn't know how to act on these cases.

Table 3 illustrates the differences in Spearman's correlation values between PCP and CPIDR proposition counts against example sentences with manually counted propositions in AID manual and in Turner and Greene. Since PCP was created to match AID manual proposition counts the correlation value is bigger than CPIDR's, however because CPIDR was created to match the proposition counts provided in Turner and Greene its correlation value is higher than PCP.

Table 3. Spearman's correlation between CPIDR and PCP against proposition counts provided by AID manual and Turner and Greene.

	PCP	CPIDR
AID manual	0.89	0.77
Turner & Greene	0.71	0.81

6.1 Correlations

Figure 11 shows comparison of proposition counts between PCP and Chand et al.'s manual. Since PCP was created on the basis of Chand et al.'s manual the Spearman's correlation value r is 0.89 which is respectable.

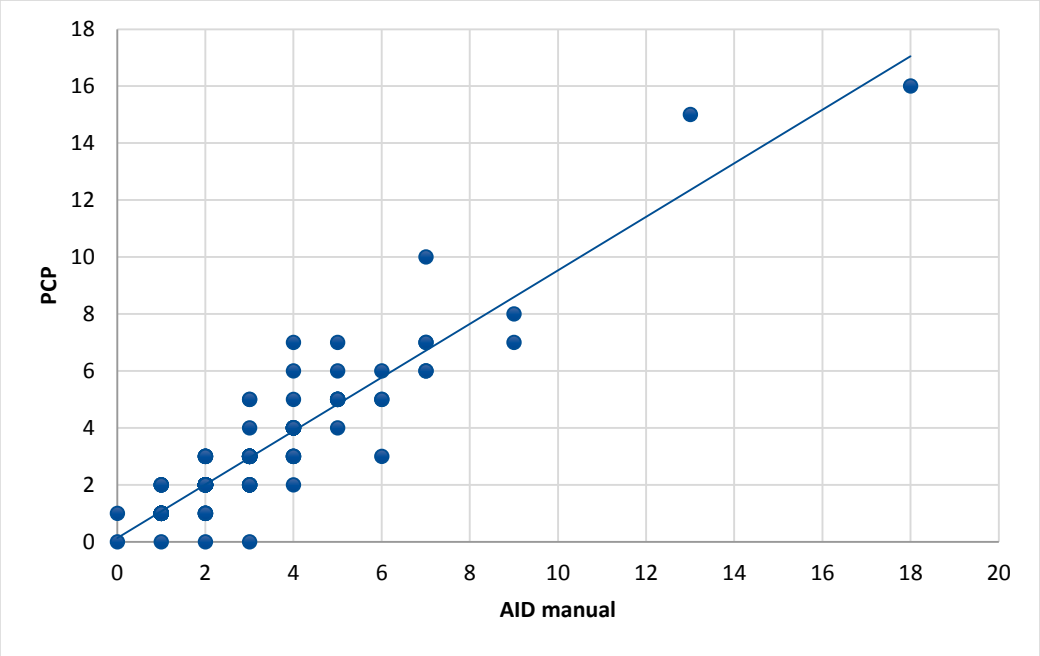

Figure 11 Correlation between my program and Chand et al.'s manual

Figure 12 shows comparison of proposition counts between my program and Turner and Greene sentences. Spearman's r value is 0.71, which is lower than on sentences from AID

manual. This is probably because in Chand et al.'s manual prepositional phrases, infinitives and other elaborative structures can function as separate ideas. So for Turner and Greene in the sentence "Fred went to Boulder" is 1 proposition but for Chand et al.'s manual there are 2 propositions.

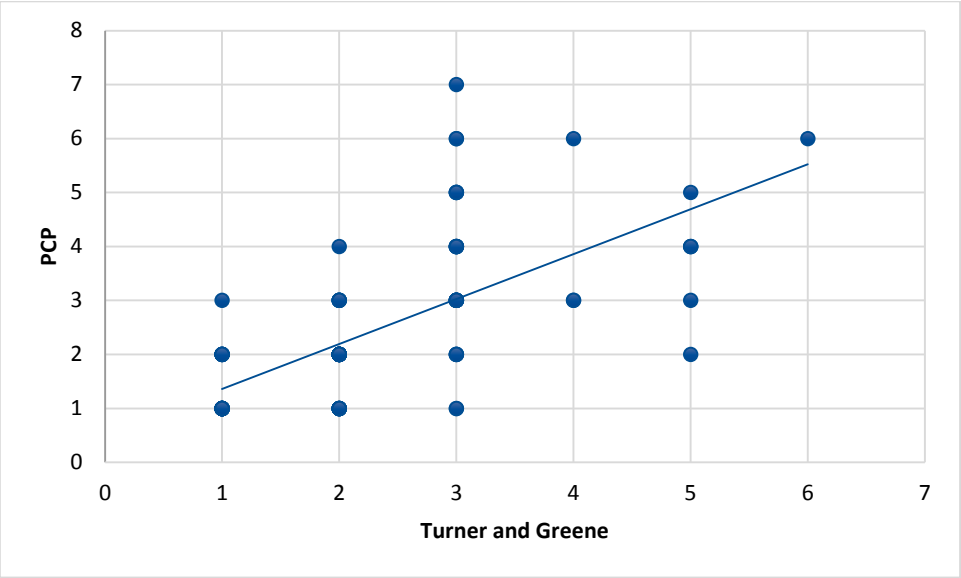

Figure 12 Correlation between my program and Chand et al.'s manual

Figure 13 shows comparison of my program and CPIDR against sentences provided by Turner and Greene. As mentioned in paragraph 3.1 CPIDR was constructed to match against propositions counts provided in Turner and Greene, but still my programs and CPIDR's spearman's correlation value r is 0.9, because the rules for finding propositions are altered a lot to more recent analysis of finding propositions count. For example according to Turner and Greene in the sentence "May not have been singing" is one propositions but in CPIDR thanks to additional rules there are two, like in my program.

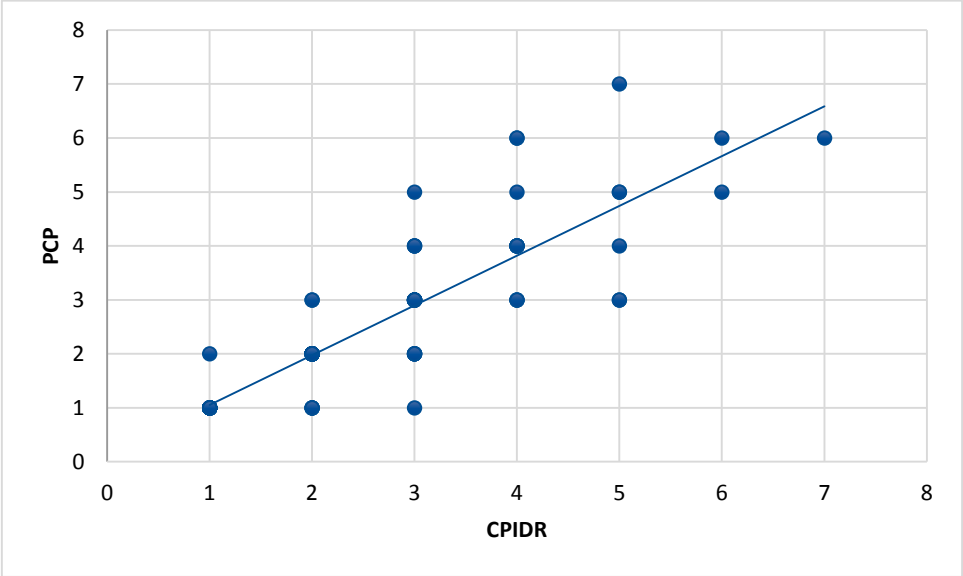

Figure 13 Correlation between my program and CPIDR for the 98 sentences provided by Turner and Greene

Figure 14 shows comparison of proposition counts between CPIDR and Turner and Greene. With Spearman's correlation value of 0.81 it is better than my program provided but that's expected because CPIDR rules were created according to Turner and Greene manual.

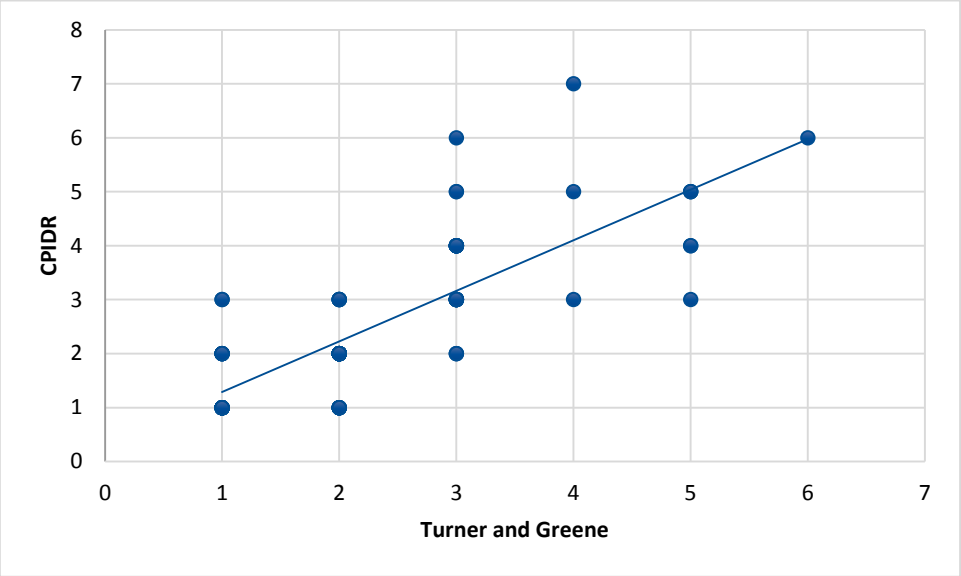

Figure 14. Correlation between CPIDR and Turner and Greene

Figure 15 shows the correlation graph between CPIDR and AID manual. With Spearman's correlation value of 0.77 it is lower than PCP's. This difference is expected, because CPIDR rules were constructed on the basis of Turner and Green but my patterns were made on the basis of AID manual. The correlations are not far apart from each other because AID manual was created on the basis of Turned and Greene with some additional exceptions and improvements.

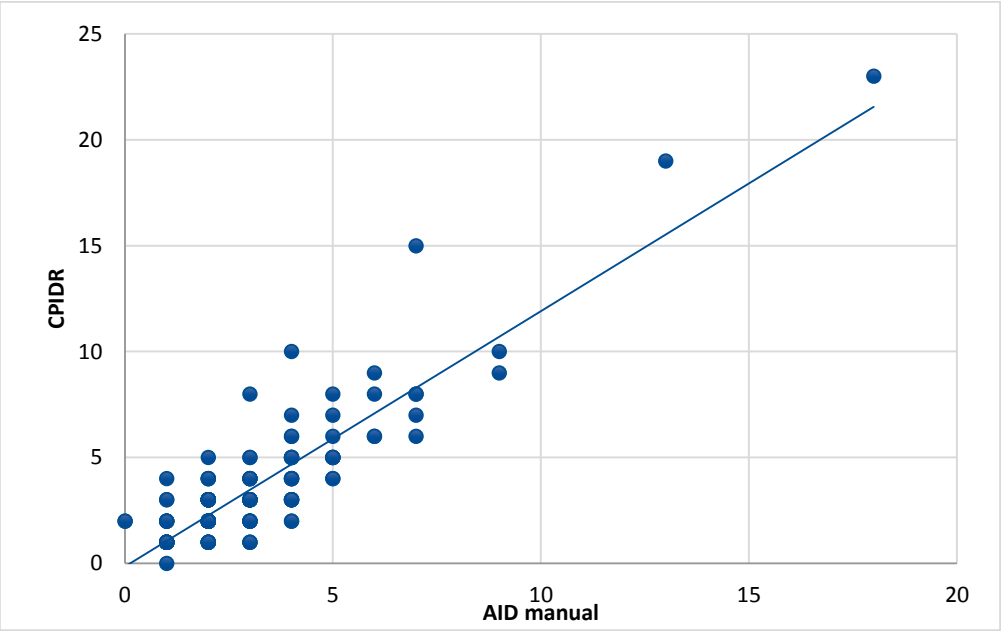

Figure 15. Correlation between CPIDR and AID manual

6.2 Patterns count

Overall I constructed 46 pattern and when tested on sentences provided by Chand et al AID manual [2], patterns match on 498 propositions. Pred1, which is the main pattern, matches 205 times what is about 41.2% out of all the propositions. Pred2 finds 86 propositions, which is about 17.3% out of all the propositions and Mod1 extracts 51 propositions what is about 10.3%. So 3 patterns out of 46 match 342 times which is 69% out of all the propositions.

In figure 16 are all the patterns with the count of how many propositions were extracted with them. The results show clear trend that there are few patterns that match many times and all other patterns extract only few special propositions. This kind of trend is very distinctive to natural language processing.

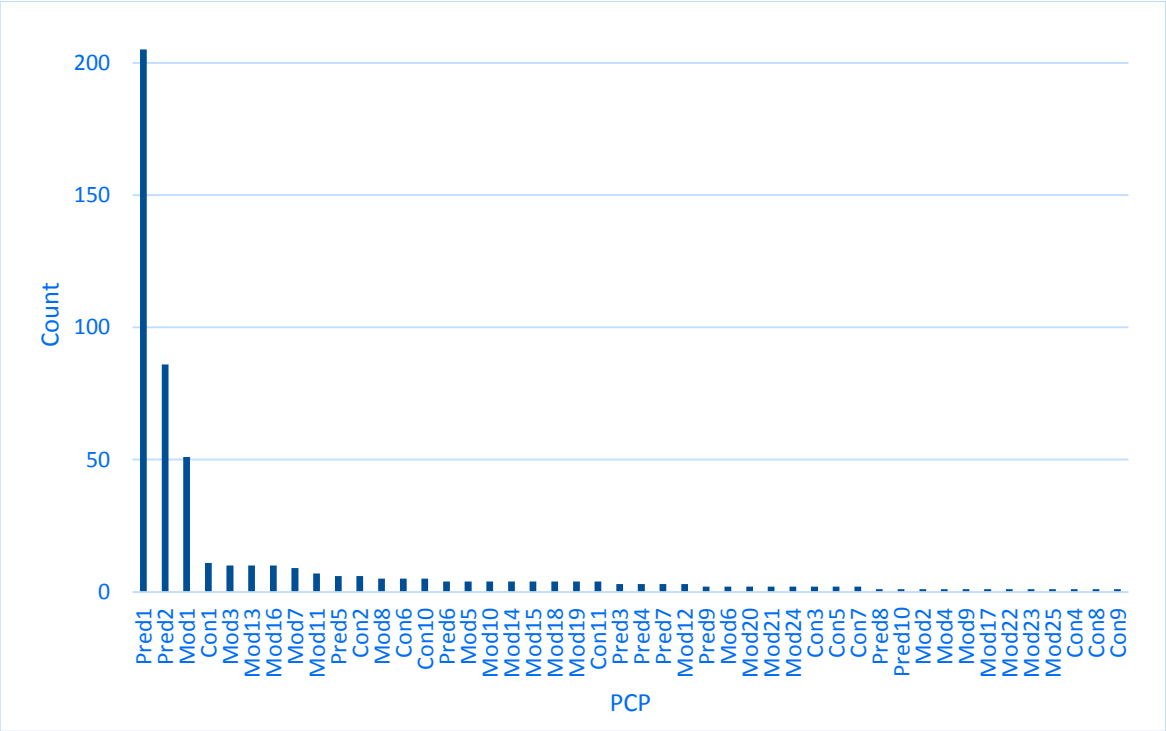

Figure 16. Patterns and the count of how many propositions was found with it

7. Conclusion

The goal of this thesis was to automatically extract propositions from dependency parse trees using patterns. The thesis shows that finding propositions using patterns is feasible solution, what shows the Spearman's correlation value of 0.89. The purpose of finding and counting propositions is that it has been found to be good measurement to relate it with readability, memory or prediction of Alzheimer's disease and doing it automatically is less time consuming than doing it manually.

To extract propositions from sentences I used patterns, which were created on the basis of AID manual. The total number of patterns I wrote is 46, where three most used patterns found about 69% of all the propositions. The Spearman's correlation value of 0.89 between PCP and AID manual shows that PCP couldn't extract all the correct propositions. One of the reasons of correlation not being higher is that AID manual has sentences that are grammatically wrong or contain nonlexical fillers. In those cases, because parser doesn't know how to process the sentence the patterns extract wrong propositions or don't extract anything. The same problem arises when parser parses a sentence differently from how AID manual processes it.

Compared to CPIDR, which uses different parser and uses POS tagger to count propositions got Spearman's correlation value of 0.77 on the sentences provided by AID manual. The lower value occurs because CPIDR created rules on the basis of Turner and Greene. PCP got Spearman's value of 0.71 when calculated on Turner and Greene sentences. The lower value occurs because AID manual uses different rules for extracting propositions, for example prepositional phrases, infinitives and other elaborative structures can function as separate ideas.

As for future work, it is possible to analyse sentence before using pattern. Analysing sentences can potentially solve the problems regarding ungrammatical sentences and sentences which have nonlexical fillers by manually or automatically fixing the sentences before trying to match patterns on them. Also it is feasible to add more patterns that could extract propositions from sentences which have special structure or are rare. It is also possible to change some patterns to make them simpler or more concrete, so that they could only extract propositions with special structure.

References

[1] A. L. V. d. Cunha, L. B. (2015). Automatic Proposition Extraction from Dependency Trees: Helping Early Prediction of Alzheimer's Disease from Narratives. *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, (pp. 127-130). Sao Carlos.

[2] V. Chand, K. B. (2010). *Analysis of Idea Density (AID): A Manual.* University of California at Davis.

[3] D. Snowdon, S. K. (1996). Linguistic ability in early life and cognitive function and Alzheimer's disease in late life: Findings from the Nun Study. *JAMA, 275*(7), 528–532.

[4] C. Brown, T. S. (2008). Automatic Measurement of Propositional Idea Density from Part-of-Speech Tagging. *Behavior research methods, 40*(2), 540-545.

[5] A. Turner, E. G. (1977). *The construction and use of a propositional text base.* Boulder: University of Colorado.

[6] D. Chen, C. M. (2014). A fast and accurate dependency parser using neural networks. *Empirical Methods in Natural Language Processing*, (pp. 740-750).

[7] W. Kintsch, J. K. (1973). Reading rate and retention as a function of the number of propositions in the base structure of sentences. *Cognitive Psychology, 5*, 257-274.

[8] E. Thorson, R. S. (1984). Viewer recall of television commercials: Prediction from the propositional structure of commercial scripts. *Journal of Marketing Research, 21*, 127-136.

[9] S. Kemper, M. T. (2001). Longitudinal change in language production: Effect of aging and dementia on grammatical complexity and propositional content. *Psychology & Aging, 16*, 600-614.

[10] D. A. Snowdon, L. H. (1999). Linguistic ability in early life and longevity: Findings from the Nun Study. Springer-Verlag, Berlin, Germany.

[11] D. A. Snowdon, L. H. (2000). Linguistic ability in early life and the neuropathology of Alzheimer's disease and cerebrovascular disease: Findings from the Nun Study. *Annals of the New York Academy of Sciences, 903*, 34-38.

[12] Nathanael Chambers, D. C.-C. (2007). Learning alignments and leveraging natural logic. *In Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (RTE '07)* (pp. 165-170). Stroudsburg: Association for Computational Linguistics.

[13] Christopher D. Manning, M. S. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Association for Computational Linguistics: System Demonstrations*, (pp. 55-60).

[14] Honnibal, M. a. (2015). An Improved Non-monotonic Transition System for Dependency Parsing. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1373-1378). Association for Computational Linguistics.

[15] Liu, H. (2004). MontyLingua: An end-to-end natural language processor with common sense. Retrieved from web.media.mit.edu/~hugo/montylingua

Appendix A

Pred1:

{}=10 ? >aux {}=12 ? >auxpass {}=11 ?>prt {}=29 >/nsubj|csubj|nsubjpass/ (!{pos:/WDT|DT/;word:/(?i)some|which/}=20 ? >/det|poss/ !{pos:/NN.*/}=21) ? >/dobj/ (!{pos:WP}=30 ? >det !{word:some}=31) ?>oprd ({}=40 ? >/det|compound/ {}=41) ? >/attr/ ({}=12 ? >nummod {}=13) ? >acomp ({}=50 ? >npadvmod {}=51) ? >xcomp ({pos:/VBG/}=60 ? >aux {}=61) ? >intj {}=70 ? >advmod ({}=80 >npadvmod {}=81)

Pred2:
!{word:/(?i)some/;pos:DT} >prep ({}=10 >/pobj|dobj|prep/ ({}=20 ? >det {word:/the|a|an/}=22 ? >compound {}=23 ? >poss ({pos:/NN.*/} >det {}=24) ? >poss {pos:/PRP.*/}=25 ? >conj !{pos:/NN.*/}=30 ? >amod {}=40))

Pred3:
{} >dative ({}=10 >/det|pobj/ {}=11)

Pred4:
{} >oprd ({pos:/NN.*/}=10 ? >det {}=11 ![>compound {}])

Pred5:
{} >xcomp ({}=12 ? >aux {}=12 >dobj ({}=10 ? >poss {}=11))

Pred6:
{} >/csubj|pcomp/ ({}=10 >dobj !{pos:WP}=20)

Pred7:
{}=10 >pcomp ({} >dobj {pos:/NN.*/}=20)

Pred8:
{} >prep ({}=10 >pcomp ({} >oprd ({}=20 >compound {}=21)))

Pred9:
{} >advcl ({}=10 >dobj ({}=20 >det {}=21) ? >aux {}=11)

Pred10:
{} >prep ({}=10 >pobj ({}=20 >amod ({} >conj {}=30)))

Mod1:
{} >/nsubj|dobj/ ({}=10 >/amod|poss/ ({pos:/NN.*|JJ/}=20 ? >case {}=30))

Mod2:
!{pos:/CD|JJ|NNP/} >advmod (!{word:then}=20 ? >neg {}=21 ![>/npadvmod|prep/ {}])

Mod3:
{} >/pobj|dobj/ ({}=20 >poss ({pos:NN}=30 ? >case {}=40 ? >conj {}=40))

Mod4:
{} >intj {}=10 >pobj ({} >nummod {}=11)

Mod5:
{} >dobj ({}=10 >nummod ({}=20 ? >amod {}))

Mod6:
{} >/prep|amod/ {word:for}=20 >/npadvmod|quantmod/ {}=30

Mod7:
!{pos:RB} >neg {}=10

Mod8:
{} >pobj ({}=10 >amod ({}=20 ![>cc {}]))

Mod9:
{} >attr ({}=10 >cc {}=20 >conj ({}=30 >quantmod {}=31))

Mod10:
{}=10 >relcl ({}=11 >/attr|acomp/ {}=20)

Mod11:
{} >attr ({}=10 >/nummod|amod/ {}=20)

Mod12:
{pos:/JJ|RB/}=10 >npadvmod ({}=11 >nummod {}=30)

Mod13:
{pos:/VB.*/} >npadvmod ({}=10 >/amod|det|nummod/ {}=20)

Mod14:
{}=10 >appos {}=20

Mod15:
{}=10 >advmod {}=11 >cc {}=20 >conj {}=30

Mod16:
{pos:JJ}=10 >advmod {}=11

Mod17:
{pos:NNP} >advmod {}=10 >prep {}=20

Mod18:
{pos:/VB|VBP/} >npadvmod {pos:/NNP|NN/}=10

Mod19:
{} >pobj ({}=10 >nummod ({}=20 ? >conj {}=40))

Mod20:
{}=10 ? >aux {}=11 >nsubj ({pos:DT} >prep ({} >pobj ({}=20 ? >det {}=21))) ? >acomp {}=30

Mod21:
{} >nsubj ({pos:DT}=20 >prep ({word:of}=30 >pobj {}=10))

Mod22:
{} >nummod ({}=10 >amod {}=20 >quantmod {}=30)

Mod23:
{} >dobj ({}=10 >det {word:some}=11)

Mod24:
{} >pobj ({}=10 >det {word:both}=20)

Mod25:
{} >dative {pos:PRP}=10

Con1:
{$} >cc {}=10 >conj (!{pos:/VBZ|JJ/} ? >advmod {}=30)

Con2:
{} >advcl ({} >mark {}=10)

Con3:
{} >preconj {}=10 >cc {}=20

Con4:
{} >agent ({}=10 >pobj ({}=20 >det {}=21))

Con5:
{}=10 >nsubj ({} >cc {} >conj {}=20)

Con6:
!{pos:IN}=10 ? >nsubj {}=20 >/dobj|pobj/ ({} >cc {} >conj {}=30 ?>poss {}=31)

Con7:
{} >prep ({}=10 >pobj ({} >conj ({pos:/NN.*/}=20 ?>compound {}=21)))

Con8:
{}=10 >conj {pos:/VB.*/}=12 >poss {}=11

Con9:
{} >pobj ({pos:CD}=10 >cc {}=20 >conj {}=30)

Con10:
{} >conj ({}=10 >dobj ({}=20 ? >det {}=21))

Con11:
{} >conj ({}=10 ? >aux {}=11 ?>acomp {}=30) >nsubj (!{pos:PRP}=20 ? >det {}=21)

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I**, Reio Laabus,**

1. herewith grant the University of Tartu a free permit (non-exclusive licence) of my thesis **Extracting semantic propositions from dependency trees**

supervised by Kairit Sirts,

1.1 to reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright;
1.2 to make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **11.05.2017**