Wrocław University of Science and Technology

HR EXCELLENCE IN RESEARCH

COMPUTER SCIENCE
Internet Engineering
II Level - MSc

Faculty of Electronics

Why should you choose Internet Engineering?

- Better job prospects

VOLVO Information Technology, intel, Nokia Siemens Networks, IBM, Microsoft

- Studies support international student exchange

erasmus

- Double diploma programmes
  - Blekinge Institute of Technology, Sweden
  - Cranfield University
  - TU Dresden, Germany

BLEKINGE INSTITUTE OF TECHNOLOGY

TECHNISCHE UNIVERSITÄT DRESDEN

- Scientific path: conferences DepCoS, ICAISC, RelStat

I semester (common with AIC)

- Discrete mathematics
  - W. Bożejko
- IT Applications:
  Electronic media in business and commerce
  - T. Walkowiak, D. Caban, M. Woda
- Information systems modelling –
  UML and service description languages
  - T. Kubik

I semester (common with AIC)

- Computer Project Management

- Research skills and methodologies

- *Elective*:
  - Signals, systems and control
  - Computer Games: Designing

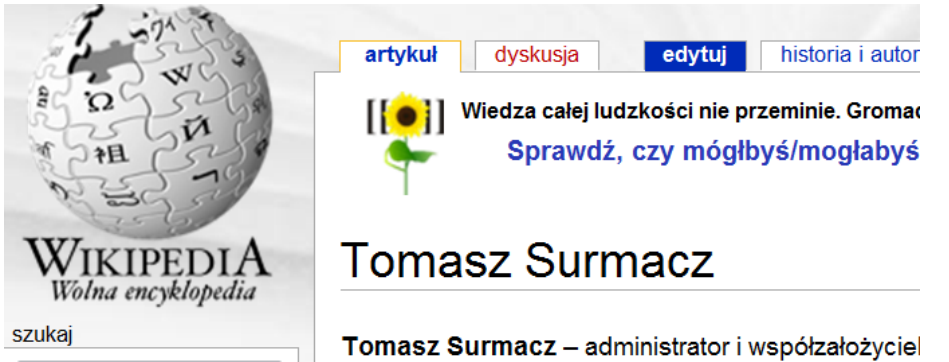
I semester (common with AIC)

- English language B2+
- Social communications
- Physics

Contact hours/week					CHS	TSW	ECTS
L	T	lab	p	s			
150	15	45	75	30	315	900	30

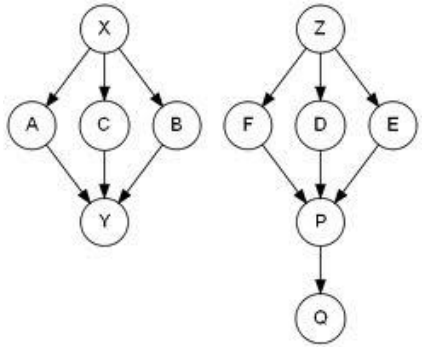
II semester

- Application programming – Java and XML technologies
  - T. Walkowiak
- Softcomputing
  - J. Mazurkiewicz
- Secure systems and networks
  - T. Surmacz



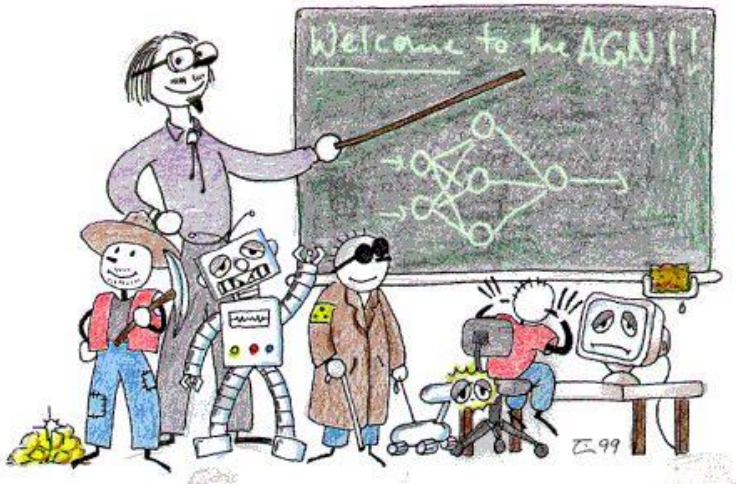
II semester

- Advanced databases
  - M. Nikodem
- Multimedia
  and computer visualisation
  - M. Woda
- Information systems analysis
  - J. Magott



II semester

- Foreign language A1



Contact hours/week					CHS	TSW	ECTS
L	T	lab	p	s			
135	60	75	60	0	330	900	30

III semester

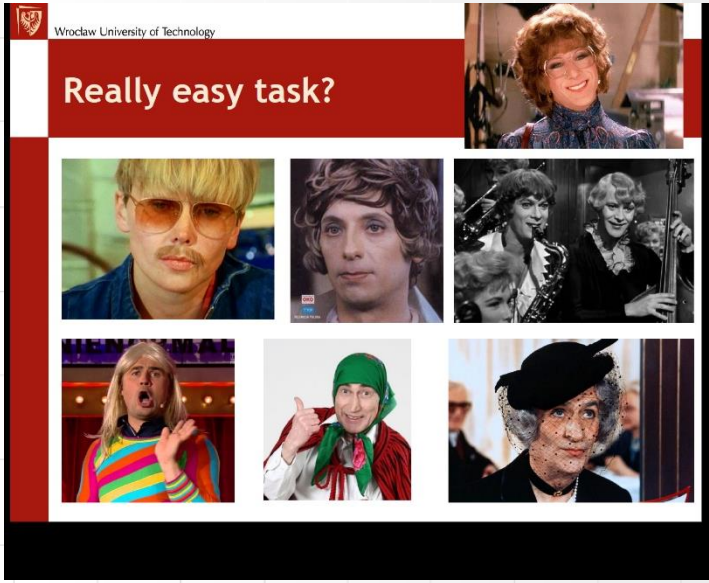
- Data mining and data warehousing
  - H. Maciejewski
- Mobile computing
  - M. Piasecki
- Final project + seminar
- Enterpreneurship

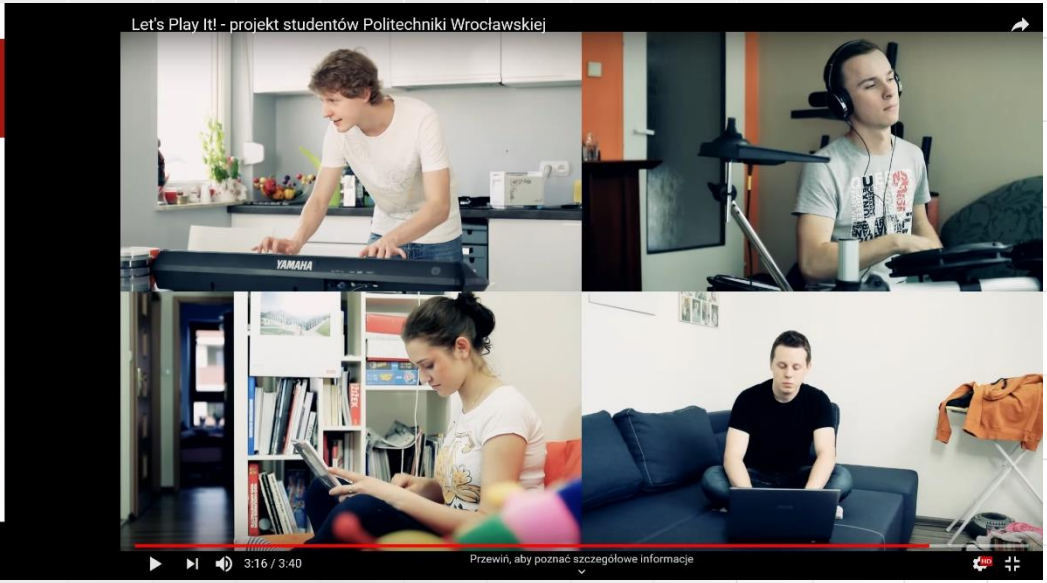
Contact hours/week					CHS	TSW	ECTS
L	T	Lab	p	s			
75	0	60	0	45	180 + P	900	30

Internet Engineering

Diploma Thesis

- ✓ Thermo-Simulation
- ✓ Let's Play It
- ✓ Recognition: faces / emotional state / gender / …
- ✓ Weather Forecasting, Bike Equipment, Data Analysis

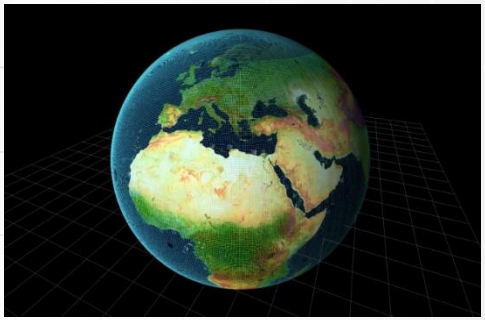
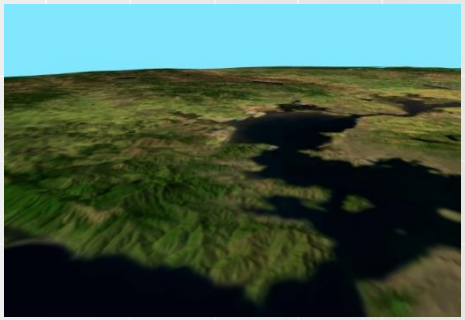
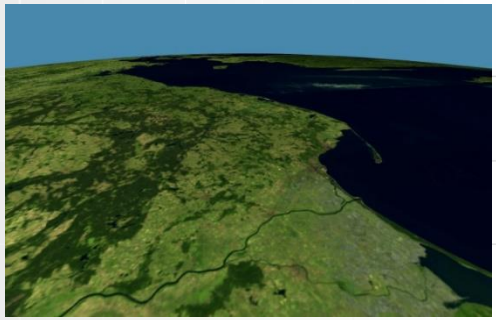
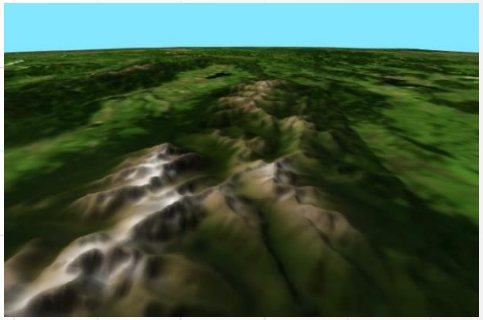
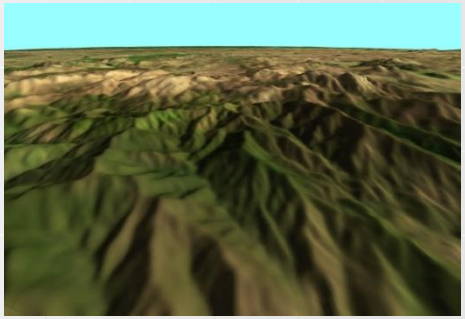
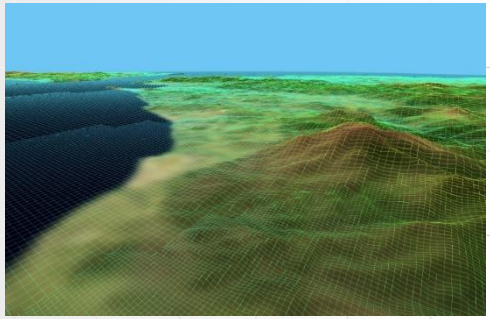




Internet Engineering

Diploma Thesis

- ✓ Earth Surface Modeling

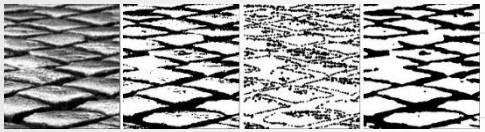
Wrocław University of Science and Technology

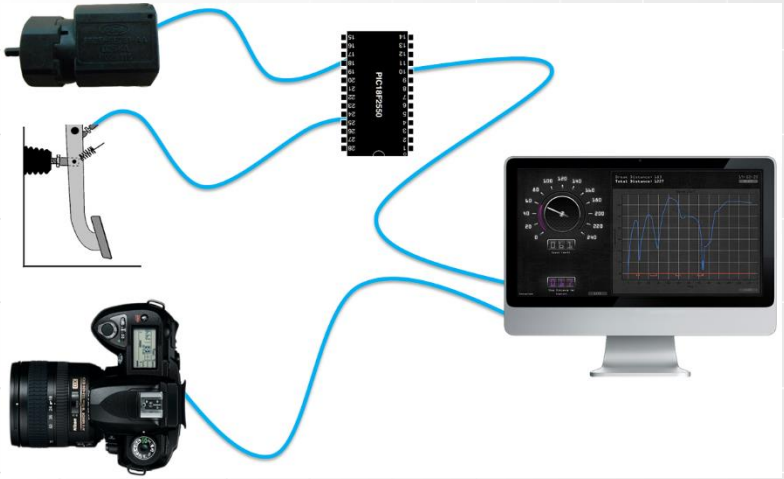
Politechnika Wrocławska

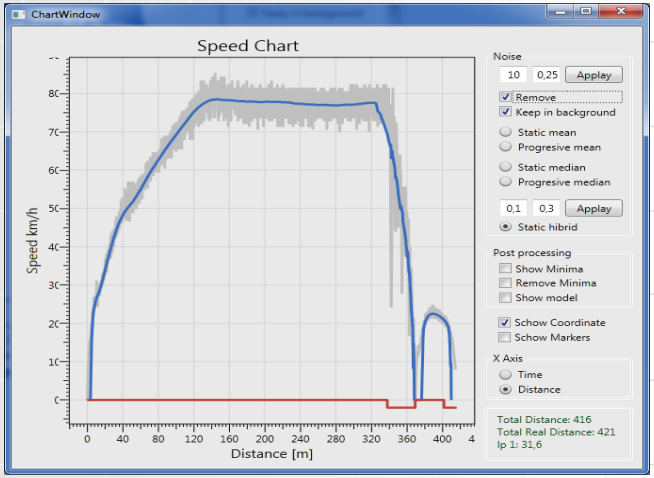
Internet Engineering

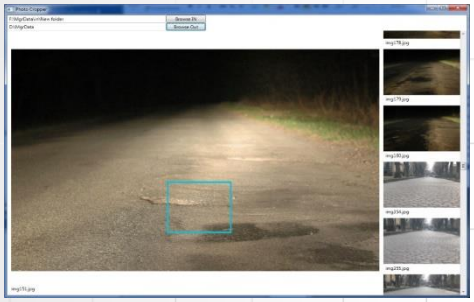
Diploma Thesis

- ✓ Automotive - Intelligent Cars



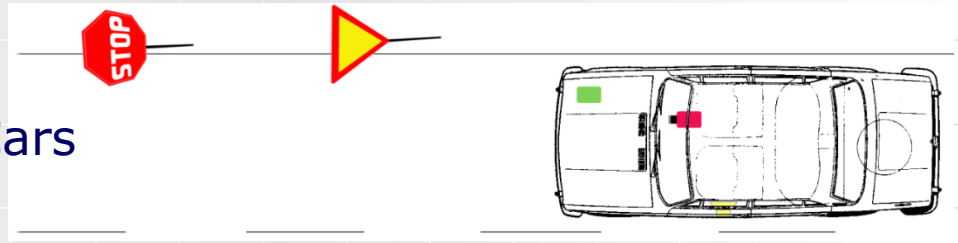




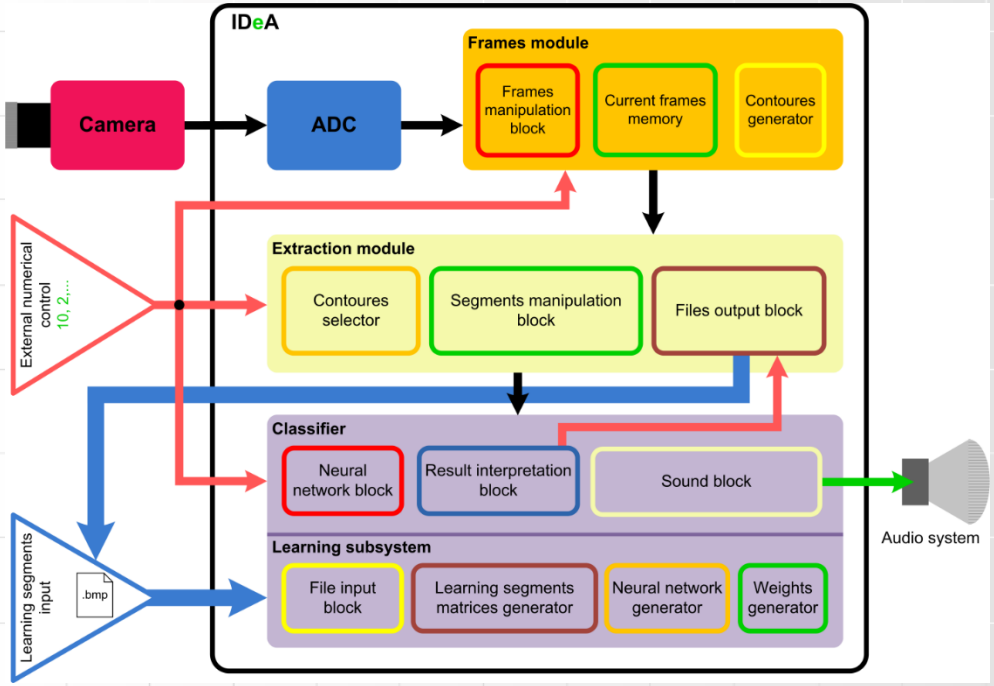


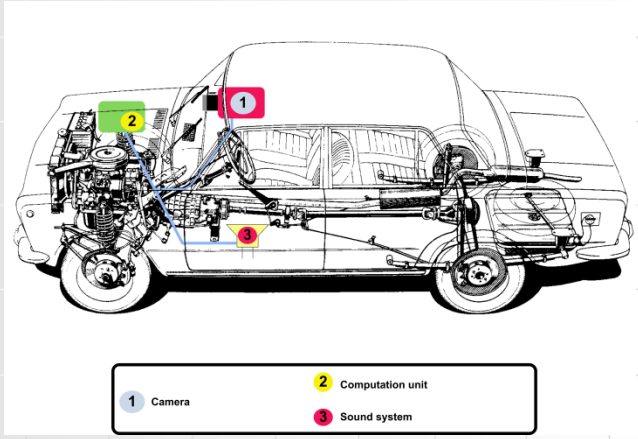
Internet Engineering

Diploma Thesis



- ✓ Automotive – Intelligent Cars





Wrocław University of Science and Technology

Politechnika Wrocławska

Internet Engineering

Diploma Thesis

- ✓ Artificial Poet

Data-driven Polish Poetry Generator

Marek Korzeniowski, Jacek Mazurkiewicz
Department of Computer Engineering, Faculty of Electronics,
Wrocław University of Science and Technology

Politechnika Wrocławska

Introduction

This work focused on a rather unusual problem – poetry generation. A complex NLG system was designed and implemented to answer a simple question: could a computer program be taught to create poems that would appear as work of a human to an average person? To that end many systems were designed in the the past, but none of them could provide an output with proper rhyme and rhythmic structures. What is more, none of them were data-driven – they usually depended on hand-crafted lexicons and sentence patterns. Finally, no similar work was conducted on Polish language – a highly fusional language – that has been chosen to be the constructed system's output.

System Overview

The poetry generation system can be divided into three main parts: the knowledge acquisition algorithm, the knowledge database and the sentence generation algorithm. A more detailed overview has been illustrated below:

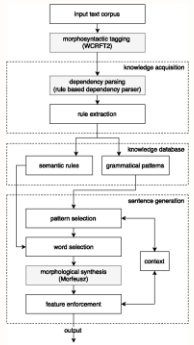


Figure 1: The poetry generation system. External modules (represented by gray fields): Morfeusz morphological analyser [4], WCRFT2 morphosyntactic tagger [3] and rule based dependency parser [1].

Knowledge Acquisition

Knowledge is acquired from an morphosyntactically tagged and dependency parsed input text corpus.
Semantic rules are created by simply extracting and saving all the parent-child pairs from the dependency trees. A single rule consists of base forms and morphological tags of two words.

Grammatical patterns are created by striping the dependency trees from words, storing only information about their grammatical structure.

Sentence Generation

The whole generation process can be described as a random search with backtracking throughout a tree of all possible word combinations. These combinations are limited by grammatical patterns and semantic rules. Example output:

I była mina i pycha dokoła grobu w agonii
skalnej, była, była noc z karku, zaś bogini mu
na przemian kotka rzekł i myślę wrócić tysiącem
serc za ślepą wieczność naszych progów.
And there was a facial expression and pride
around a grave in rockery agony, it was, it was
the night from one's neck, whereas the goddess
alternately created a kitten from words and I
think about returning into blind eternity of our
doorsteps by means of a thousand hearts.

Mąż każdego cwału jącego stanu śmiał ku mnie
wyjść ciekawym okiem!
A husband of each galloping state had the in-
solence to leave through a curious window and
walk in my direction!

A complete implementation of the generator using 320k word text corpus worked with the speed of 2k words per second. The computation was performed on a single core of a standard 3GHz Intel i7 processor.

Feature Enforcement

A special feature enforcement algorithm was created to ensure rhythm and rhymes in the output. Its construction was based on the following reasoning: "If we make the generator really fast we could simply wait until it produces results with proper rhythm and rhymes".

Context

A list of recently used nouns is kept in memory to provide the output text with an illusion of context. Subjects inserted into patterns are chosen from this list.

Rhythm

Ensuring proper rhythm in Polish is surprisingly easy when compared with other languages because of constant accent. Every word is accented in the exactly same way. When writing a poem, one has only to ensure that each verse has the same amount of syllables. Example output:

Mgła w oku okna boleje.
A fog suffers in an eyes's window.
Znowu łzawy nie wyrzucam.
Again I do not throw lava away.
Strącam w ognie, serca gryzę,
I throw into flames, I gnaw hearts,
Dla mody gaszę, nie żywię!
For fashion I rebuke, I do not nourish!

Rhymes

To force a rhyme the last word of each even verse is replaced by a random word with a equal morphological tag and a correct rhyme. Semantic rules are ignored during this operation. Example output:

Skrę wydałem, skra gorzała
A spark I gave, a spark has blazed
I gorzała jej czekała!
And it's booze has hazed!
Ja nie wyjmę, skra rozdmucha,
I will not remove, the spark will scatter,
Zbożę za mną nie wybucha!
The grain behind me does not shatter!
Już setna skra się rozlewa,
Already the hundredth spark is spilling,
A skra potem się przelewa!
And then a spark starts swilling!

To make the translation more accurate the last words of even verses were randomly altered to form rhymes in English.

Results

Finding a good metric to measure the quality of the generator's output proved difficult. A decision was made to conduct a public survey. It was composed of sixteen short poetry fragments: ten generated and six randomly chosen from classic Polish poems. The task was to determine which fragments were computer generated.
Eighty-six people were surveyed. Each person was scored on a scale of sixteen – one point for each correctly answered question. The average score was 11.1 (70% fragments correctly classified).
Only one generated fragment had the ratio of incorrect answers above 50% – 50 out of 86 participants (58%) had classified the following fragment as work of a famous Polish poet:

Słowo pali, słowo pędzi,
The word burns, the word dashes,
Słowo się pode mną szczęści.
The word lavishes under me.
Słowo ciska naokół, gdzie czci,
The word hurls around wherever it reveres,
Wiecznie zbywam, i słowo lśni.
Eternally I put it off and it gleams.

Summary and conclusions

The survey results seem to be surprisingly good, but it needs to be mentioned that the fragments used in it were very short. Recognising human poetry in longer portions of text would be a much easier task as the generator output has no coherent meaning.
Also, the achieved output semantic correctness was lower than expected. Simple two-word rules were too weak to create meaningful and complex sentences. Perhaps the quickly growing Polish wordnet "Słowosieć" would provide means to achieve better results [2].

References

[1] Korzeniowski, M., Mazurkiewicz, J.: Rule based dependency parser for polish language. In: Artificial Intelligence and Soft Computing, ICAISC 2017 (June)
[2] Piasecki, M.: pWordNet (Słowosieć) (2008), http://hdl.handle.net/11321/43
[3] Radziszewski, A., Warzocha, R.: WCRFT2. CLARIN-PL digital repository (2014), http://hdl.handle.net/11321/36
[4] Woliński, M.: Morfeusz—a practical tool for the morphological analysis of polish. In: Intelligent information processing and web mining, pp. 511–520. Springer (2006)

Rule Based Dependency Parser for Polish Language

Marek Korzeniowski, Jacek Mazurkiewicz
Department of Computer Engineering, Faculty of Electronics,
Wrocław University of Science and Technology

Politechnika Wrocławska

Introduction

Currently there are two leading dependency parsers available for Polish language: Świgra and "Polish Dependency Parser" [4, 5]. The first one is based on a formal grammar model of Polish language and the second represents a data-driven approach.
These solutions work well for multiple applications but problems arise when one tries to analyse non-standard texts, like poetry. For complex sentences Świgra often does not return results in reasonable time or does not return them at all. "Polish Dependency Parser" used on texts drastically different from the ones it was trained on returns gibberish.
We propose a completely different approach to dependency parsing: a method based on a chain of simple heuristic rules, operating on words and their morphological tags. Each rule removes a word from the input and attaches it to a different word – effectively joining them into a parent-child pair. The proposed parser can analyse any kind of grammatically correct texts and return correctly parsed sentences only.

Parts of Speech Archetypes

The morphological tagset used in this work defines over thirty parts of speech [3]. The following archetypes were introduced to reduce the complexity of the parser:

- noun (*subst, depr, ppron12, ppron3, siebie* and *ger*) representing parts of speech that can become the subject of a sentence,
- verb (*fin, bedzie, praet, impt, imps* and *winien*) representing parts of speech that can become the predicate,
- adjective (*num, numcol, adj, pact* and *ppact*) representing parts of speech that describe nouns,
- adverb (*adja, adjp, adjc, advc, infl, pcon, ppant* and *pred*) representing parts of speech that describe verbs.

Parts of speech not included in the archetypes are simply ignored and they are not included in the output.

Dependency Trees

The output trees are quite crude and carry much less information compared with trees returned by other parsers. A short summarization of their structure:

- The root is always a verb and can have child nodes being nouns, adverbs and prepositions.
- Prepositions always have one child which needs to be a noun.
- Adverbs can have children being other adverbs or adjectives.
- Nouns can have children being other nouns, adjectives or prepositions.
- Adjectives can have a single child (also a adjective) if they represent a compound numeral.

The trees are rooted in verbs as in Polish compound sentences can easily be devised into sub-sentences contacting exactly one verb being the predicate. Verb-less sentences are discarded as unrecognised.

Parsing Rule Chain

The complete parsing rule-chain is summarized below:

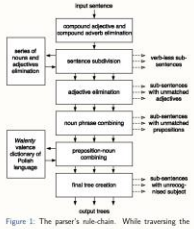


Figure 1: The parser's rule-chain. While traversing the rule-chain, sub-sentences can get discarded as unrecognised for numerous reasons – shown on the diagram in form of dashed lines.

Input data

The parser accepts as input morphosyntactically tagged text divided into sentences. This data is passed through the rule-chain, where each stage joins different grammatical constructions into sub-trees.

Compound Adjective and Compound Adverb Elimination

This stage joins compound numerals and adverb-adjective / adverb-adverb combinations. For example phrases like *"niewiele szybciej"* ("slightly faster") and *"niesamowicie zielony"* ("amazingly green") would be transformed into sub-trees rooted in *"szybciej"* and *"zielony"*.

Sentence Subdivision

A specially designed state machine finds series of nouns and adjectives and joins them into sub-trees. For example the phrase *"Znalazłem psa i kota"* ("I found a cat and a dog") would be replaced by a sub-tree rooted in a placeholder plural noun with appropriately derived case and gender.
Without series of nouns and adjective a sentence can be easily split into sub-sentences on every comma and conjunction. Each sub-sentence will be from now on processed independently and will produce a separate tree on the output.
There are some rare cases where the state machine will incorrectly detect series of nouns, for example *"Znalazłem psa i kota tym psem poszczułem"*.

Adjective Elimination

The previous stage eliminates practically all adjectives by connecting them with nouns. In some cases adjectives can appear not directly before the nouns described by them. For example: *"W czarodziejskiej cip celwiezłem wiezy"* ("In magicians you I visited tower"). The third step of the parsing rule chain was created to handle such cases.

Preposition-Noun Combining

This stage simply connects preposition with nouns into sub-trees rooted in the preposition.

Noun Phrase Combining

In Polish every noun in a sentence can be connected with the predicate or with an adjacent noun. The connection can be direct or via a preposition. If the two nouns are separated by a preposition there is no simple way to determine if they should be connected or not. This problem was partially solved using Walenty – a valence dictionary of Polish language [1].

Final Tree Creation

The predicate is placed in the root of the output tree. All adverbs, preposition and nouns left in the input are added as its children. All other words are discarded.

Results

To test the created parser a corpus of Polish poetry was prepared. It was build from the work of classical Polish poets such as Adam Mickiewicz and Bolesław Leśmian. An initial preprocessing stage was applied to remove or replace all characters not being a letter, comma, punctuation mark, question mark or an exclamation mark. The text was tagged using the WCRFT2 tagger [2]. In 24219 input sentences 71051 sub-sentences were found and:

- 36052 were accepted,
- 27882 were discarded as verb-less,
- 3670 were discarded in the adjective elimination step,
- 3448 were discarded for other reasons.

The parsing took 0.45 seconds on a single 3GHz core of an Intel i7 processor. 4253 sentences were discarded before parsing because of words unknown to the tagger.

Summary and conclusions

This paper proposes a new approach to dependency parsing. A parser has been implemented and tested on Polish poetry – texts that are mostly unparsable by current state-of-the-art dependency parsers.
The largest drawback of the parser is the lack of support for verb-less sentences which leads to approximately 40% loss of the test corpus. This problem should be resolved before the parser can be considered as a fully functional tool. However, even in its current state it can be used for crude dependency analysing of texts which so far could not have been analysed at all.
Even though the described solution was created strictly for Polish, it could be applied to other languages. Especially those from the Slavic group, which prove to be hard to formalize. The parser's rule-chain would need to be modified to accommodate the target grammar, but the approach described in this paper would remain the same.

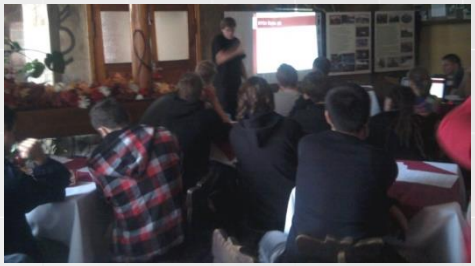
References

[1] Andrejewicz, J., et al.: Walenty. CLARIN-PL digital repository (2016), http://hdl.handle.net/11321/251
[2] Radziszewski, A., Warzocha, R.: WCRFT2. CLARIN-PL digital repository (2014), http://hdl.handle.net/11321/36
[3] Woliński, M.: System znaczników morfosyntaktycznych w korpusie ipi pan. Polonica XXIII-XXIII pp. 39–55 (2003)
[4] Woliński, M.: Swigra. CLARIN-PL digital repository (2016), http://hdl.handle.net/11321/258
[5] Wróblewska, A.: Polish Dependency Parser Trained on an Automatically Induced Dependency Bank. Ph.D. dissertation, Institute of Computer Science, Polish Academy of Sciences, Warsaw (2014)

Internet Engineering

Seminars & Tours

- Jarnołtówek, Bystrzyca Kłodzka



- CeBIT - Hannover, Ślęża, VW - Dresden

Internet Engineering

Santa Claus!

- 6th December – of course!

Internet Engineering

Not Only University ;-)

Internet Engineering

Once More – Thank You!

- Dariusz.Caban@pwr.edu.pl
  Jacek.Mazurkiewicz@pwr.edu.pl

http://www.zsk.iiar.pwr.wroc.pl/zsk/dyd/