Course title: ***Algorithms and programs for automatic text analysis***
Lecturer: ***ViatcheslavYatsko***, Doctor of Science, Full Professor, Head of the Computational linguistics laboratory (CLL) http://vetsky.narod2.ru/
Target audience: ***Students of Linguistics / Computational Linguistics / Information science / Computer science***

Introduction

This syllabus describes main theoretical notions to be studied by the students; presents lessons' plans and laboratory assignments; formulates main requirements for students' knowledge and skills, which they are supposed to acquire on completing the course.

A specific feature of this course is that it describes methodologies for creating linguistic software and automatic and semi-automatic creation of dictionaries and ontologies. It is based on personal experience of the lecturer in developing linguistic algorithms and programs. The algorithms and programs are studied according to the levels of language system, with which they are correlated: morphological, lexical, syntactic, and discursive.

The students will use linguistic software developed at the CLL and other related software designed for Windows platform. They are supposed to have skills in working with MS Office programs (Excel, Word). All the software is distributed as freeware.

Another peculiarity is that the course is not aimed at "reading" lectures. Presentation of theoretical material is immediately supported by practical assignments, doing which the students learn how to use and develop linguistic software, how to apply various laws to analyze text data, how to perform computations according to some formulae. All lessons are conducted in a computer class with individual PCs, on which the students do laboratory assignments, read materials, and analyze illustrations. The students are graded after each lesson; at the end of each lesson they submit the results of their work to the lecturer. This syllabus focuses on 9 sample lessons, but there is no problem to add more lessons adapting them to various audiences. Duration of each lesson is 80-90 minutes.

The course comprises three main parts. The first part deals with linguistic algorithms and programs; the second part concentrates on lexicographic recourses used to support the functioning of linguistic programs; the third part focuses on the laws that underlie the subject field of natural language processing.

Requirements for students' knowledge

On completing the course

the students will know:
- How to create a stemmer, evaluate strength of stemmers, identify stemming errors.
- How to create a POS tagger and differentiate between rule-based and probabilistic taggers.
- How to create a tokenizer taking into account abbreviations, set expressions, proper names.
- How to perform computations using term-weighting formulae and create reference dictionaries.
- How to recognize and calculate the number of n-grams.
- How to perform text splitting using a neuronet and deductive-inverse architecture.
- How to apply Y-interpretation of Bradford's law to generate a reference dictionary and calculate threshold values.
- How to apply Zipf's law to analyze texts and determine the degree of crisis development.
- How to evaluate linguistic software.

the students will be aware of:
- The structure of natural language processing subject field and main trends in speech processing.
- Criteria for classification of text processing programs and algorithms.
- Differences between lemmatization and stemming.
- The structure of derivational and linear grammars, which underlie the functioning of chunkers and parsers.
- Specific features of anaphora and co-reference resolution algorithms.
- Trends in discursive analysis, semantic features in rhetorical structure theory and its application for text modeling/
- The structure of thesauri and ontologies and the role in support of text mining systems.
- Shannon- Hartley law and its applications for text analysis.

Lessons plans and laboratory assignments

Lesson 1

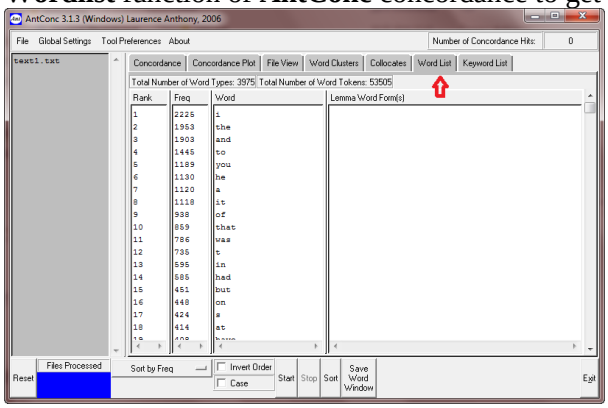
Natural language processing as a subject field. Speech recognition and text recognition. Main trends in speech recognition: identification of personal characteristics; control of technological objects; answering machines. Classification of text processing systems according to novelty of output, communication type, level of language system; differences between processing of monologic and dialogic texts (chats, blogs, forums); differences between information retrieval and text mining [**1**].

Linguistic programs and algorithms on the morphological level of language system. Stemming as a fundamental algorithm underlying information retrieval [**2**]. The notion of a stem in NLP. Dictionary-based and algorithmic stemmers, their advantages and disadvantages. Overstemming and understemming errors [**3**]. Strength of a stemmer. Y-stemmer and co-relation between suffixes and endings and parts-of-speech.

***Laboratory assignment* No 1.**

Aim: *Evaluation of strengths of Y-stemmer and M.Porter's stemmer.*
Source data: *A text*
Software: *AntConc, Y-stemmer* (developed at the CLL), *Porter's stemmer*

1) Use two stemmers to process the source text. Save results (stems of words in the text) in *.txt* files. Use **Wordlist** function of **AntConc** concordance to get data needed for calculations according to the formulas.



(1) ICF – Index compression factor

$$ICF = \frac{(N - S)}{N}$$

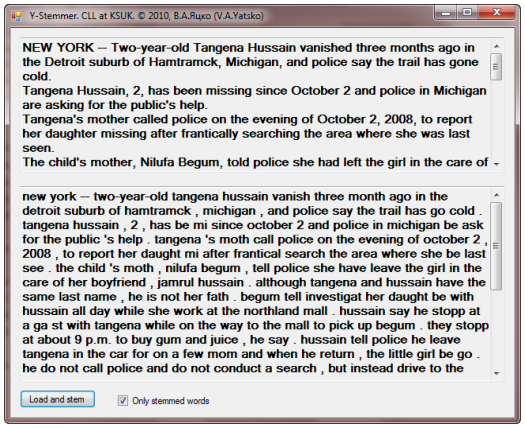
(2) Mean number of words per conflation class

$$MWC = \frac{N}{S}$$

(3) Probability index compression

$$PIC = \frac{T - (N + S)}{T}$$

where N is the number of unique tokens (word types) in the source text before stemming; S – the number unique tokens after stemming (calculated separately for output of each of the stemmers); T is total number of tokens in the source text.

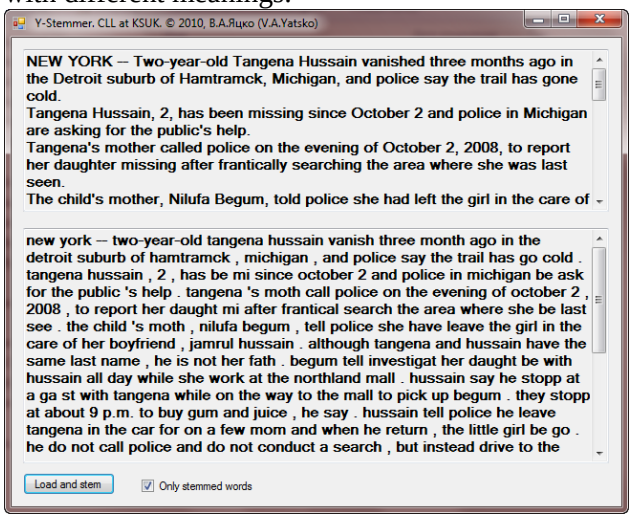


2) Calculate difference between scores of stemmers obtained from each formula

3) Sum up the differences to find out which stemmer is heavier. Explain the result.

***Laboratory assignment* No 2.**
Aim: *Evaluation of quality of Y-stemmer and Paice/Husk (Lancaster) stemmer*.
Source data: *A text*
Software: *Stemmer testing software* (developed at the CLL).

1) Find and indicated cases of overstemming (O) and understemming (U) using stemmer testing software. Note that a stem in NLP differs from its interpretation in linguistics. For example *daughter* stemmed as *daught* cannot me regarded as a mistake, since such a stem aggregates proper word forms (*daughter*, *daughters*). And *mother* stemmed as *moth* is an overstemming mistake because *moth* aggregates word forms with different meanings.

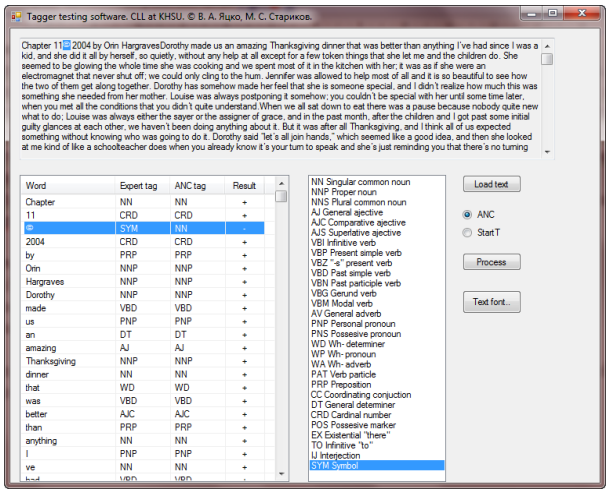


2) On completing error recognition click *Process* button and copy the results to MS Excel. Calculate the number of *U* and *O* mistakes and divide the result by the total number of tokens to find the quality score.

Lesson 2
Stemming and lemmatization. A.Kilgariff's lemmatized and unlemmatized BNC lists [**4**].
Programs and algorithms used on the lexical level of language system. Lexical decomposition and the notion of a token. Differences between words and tokens. Problems in lexical decomposition; recognition of abbreviations, set expressions, initials, dates [**5**].
Annotation with part-of-speech tags; the notion of a tag. Rule-based and probabilistic taggers. Bidirectional inference algorithm. E.Brill and a methodology for creating a rule-based tagger. A generalized architecture of a POS tagger [**6**].
Annotating texts with semantic roles and cognitive roles. The use of semantic annotation for factographic searches [**1**]. Cognitive roles in text mining [**7**].

***Laboratory assignment* No 3.**
Aim: *Evaluation of quality of StarTagger and ANC tagger*.
Source data: *A text*
Software: *Tagger testing software* (developed at the CLL).

- Open the text in the *test_tag.exe* program. Check the tagger you want to evaluate (ANC tagger is a default option). In case you find a mistake (incorrect tag) select the row and double click the correct tag in the right section to substitute the incorrect tag. Check the results of the other tagger in the same way.



When you are finished click *Process* button. Copy the results to MS Excel to calculate the quality of each stemmer dividing the result of each tagger by the total number of tokens.

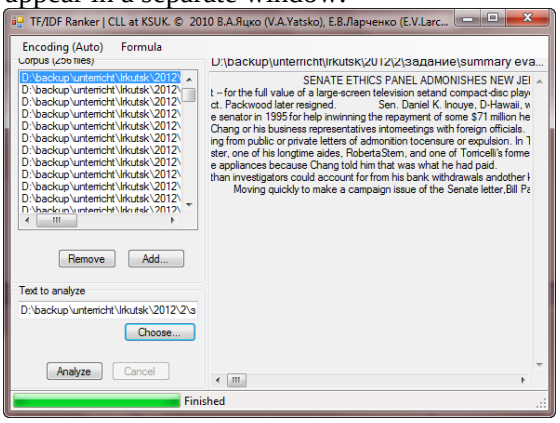
Lesson 3

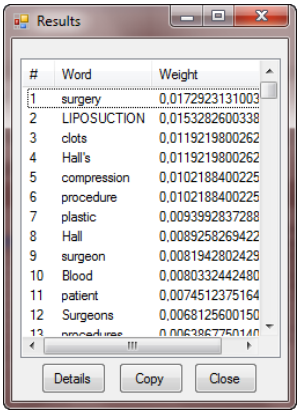
Term weighting algorithms and weighting filters; the notions of a term-weight and ranking. Inter-textual and intra-textual methods of term weighting. Raw counts and probability values. TF*IDF algorithm and opportunities for filtering stopwods, texts classification and categorization. Problems in creating a reference corpus. Modifications of TF*IDF formula. Symmetric term weighting and automatic text summarization.

***Laboratory assignment* No 4.**

Aim: *Creation of a reference dictionary*.
Source data: *A source text, a reference text corpus*
Software: *TF*IDF Ranker* (developed at the CLL).

1) Open *TF*IDF Ranker*. Select a modified version of the formula in the *Formula* menu item.
Open the text to analyze clicking *Choose* button.
Click *Add* button to add a reference text corpus. The list of texts will appear in the upper section.
Click *Analyze* button. The result (ranked list of words in the text under analysis with TF*IDF weights) will appear in a separate window.





2) Copy the ranked list and insert it in the Excel file. Sum up all weights and divide the sum by the number of tokens in the text under analysis to get the mean arithmetic value. Ignore tokens with zero and exponential weights. Select from the ranked list all words whose weight exceeds the mean arithmetic. These words will constitute the reference dictionary.

***Laboratory assignment* No 5.**

Aim: *Application of symmetric term-weighting for text summarization*.
Source data: *A text, a reference dictionary*

(1) *The basis for Western literacy was the invention of alphabetic writing by the Greeks.* **(2)** *Around 1100 B.C. the Phoenicians invented a syllabary, a writing system representing spoken syllables.* **(3)** *It is conjectured that the impetus for the Phoenician invention was probably commerce.* **(4)** *The Greeks, building on the syllabary, developed alphabetic writing where the written symbols represent meaningful sounds (phonemes) of the language*

1) Read the text given above. Take notice of the underlined words: they constitute a refence dictionary for this text.
2) Calculate functional weight for each sentence. This weight = number of connections of the given sentence with the other sentences. Number of connections of the given sentence = number of repetitions of stems from the reference dictionary (underlined). Sentence connections are calculated in accordance with the symmetry principle*: If a sentence A has N connections with a sentences B, then the sentence B has the same connections with the sentence A.*
Fill in the table:

Table 1. Sentences connections

Sentence	Connections	Connecting words	Number of connections (sentence weight)
(1)	(1)–(2)		
	(1)–(4)		
(2)			
(3)			
(4)			

2. Make up a summary of the text selecting sentences, whose weight exceeds the mean arithmetic value.
3. Have a look at the sentence with the highest number of connections and at that one with the smallest number of connections. Try to find linguistic evidence for the former to be really important for the text, and for the latter to be insignificant for the meaning of the text.

Lesson 4.

Syntactic level of language system. The notion of *n-gram*: *bigram, trigram, tetragram* [**8**]. Recognition of *n-grams* in the text; $ng_{(s)} = w_{i-(n-1)}, w_{i-(n-2)}\ldots w_{i-(n-n)}$, where w_i is the sequence number of *n-gram*, beginning with *bigrams*. Significance of the *n-grams* analysis for foreign language teaching and automatic text classification.
The notion of syntactic parsing. Types of phrases and phrase structure rules [**9**]. Recognition of hierarchical structure of a sentence. *Lexparser* (*Stanford parser*) [**10**]. The significance of syntactic parsing for modeling of text structure and machine translation.
Syntactic chunking and chunkers. Noun-phrase chankers and their importance for representing text content.

***Laboratory assignment* No 6.**

Aim: *Construction of derivation terees for ambiguous sentences*.
Source data: *Ambiguous sentences*
Software: *Lexparser*.

1) Open *Lexparser* shell and load the English parser (*English_factored*.ser file in the program's directory).
2) Anlalyze each sentence below with the help of *Lexparser*. Make screenshots of the results.
– *John saw the student with a telescope*
– *Visiting relatives can be boring*
– *I saw her duck*
– *I forgot how good beer tastes*
3). Since the sentences are ambiguous they admit of two different interpretations. Draw derivation trees for the interpretations which haven't been given by *Lexparser*. Explain why the parser has chosen its variant of interpretation.

***Laboratory assignment* No 7.**

Aim: *Demonstration of importance of noun phrases and NP chuncking* .
Source data: *A text, a reference dictionary*

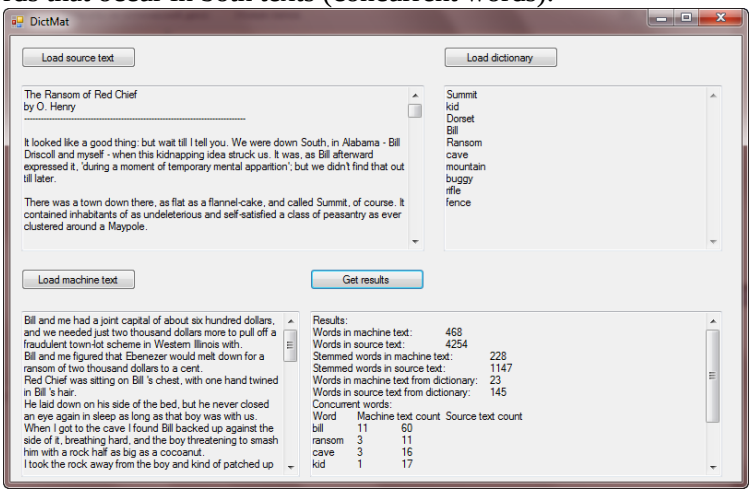
1) Go back to *Laboratory assignment* No 5. Make up a separate list of word stems in the reference dictionary (underlined in the text). Exclude from the list word stems that are not nouns. Only noun stems must be left.
2) Revise Table 1 and calculate sentence connections taking into account only connections through noun stems.

3) Analyze the result. In what way does it differ from the previous result when all words in the dictionary were counted?

Lesson 5.
Syntactic splitting as a fundamental algorithm of text decomposition. Problems in sentence recognition and typical errors. The notion of deductive-inverse architecture of text decomposition and its specific features. Possibilities for the use of neural networks for text decomposition.
Clause decomposition and the notion of a clause [**11**]. Significance of clause decomposition for modeling of logical and semantic structure of texts. Linear grammar and algorithms for recognition of noun phrases and verb phrases.

***Laboratory assignment* No 8.**
Aim: *Evaluation of quality of summaries with the help of reference dictionary..*
Source data: *Source text, summaries of source text*
Software: *Subject Search Summarizer* (*SSS*), *Copernic Summarizer*, *Dictionary matching program* (*DicMat*) (developed at the CLL).

1) Get machine summaries of the source text using *SSS* and *Copernic Summarizer*.
2) Go back to *Laboratory assignment* No 4 to use the reference dictionary.
3) Open *DicMat* and load to it the reference dictionary, source text, machine summaries, one by one. In the *Results* section of *DicMat* you will get data about the total number of tokens in two nexts and raw counts of words that occur in both texts (concurrent words).



4) For each concurrent word in the source text and in the summaries calculate probability values. The value of the word in the source text will represent a reference value.
5) Calculate **absolute** differences between probabilistic value of each word in the source text and in the summaries.
6) Sum up the differences to get a score for each summary.
7) Compare scores of summaries to estimate their quality. The less is the score the better is the summary.

Lesson 6.
Discursive text analysis. The notion of discourse in NLP. Approaches to the analysis of discourse semantic structure; W.Mann's conception, nuclear and satellite components of discourse spans [**12**].
Anaphora and the problem of anaphora resolution; importance of anaphora resolution for information retrieval, text summarization and mining. Algorithms for anaphora resolution; global-discursive and statistical approaches **[13].** Discrimination and identification rules in statistical approach.

***Laboratory assignment* No 9.**

Aim: *Analysis and understanding of text's semantic structure*
Source data: *A newspaper text*
Software: RST TOOL

1) Open RST TOOL. Use the *import text file* function to load the text under analysis.
2) Use the *Segment* module to break the text into clauses.
3) Click *Relations* module and load *ClassicMT.rel* relations list.
4) Open *Structurer* module. You will see a row of clauses separated from each other.

5) Set relations between clauses. Select a satellite component and move the mouse pointer to the nucleus. A list of relations will pop up. Click on one of the relations. Note that the structure of the text under analysis should be hierarchical.
6) Customize the appearance of the diagram using *appearance options*.
7) Capture the diagram to clipboard and insert into your .doc file.
8) Explain the hierarchical structure of the text.

Lesson 7.

Lexicographic resources for support of automatic text processing; dictionaries, thesauri, ontologies. WordNet as a lexical database [**14**]; structural-semantic relations between words; different levels of synonymy. Formal ontologies and linguistic ontologies. Functional relations between ontology units [**15**]. How to create an ontology for opinion mining; syntactic and semantic terms; sentiments intensity; grammars.
Laws underlying the NLP subject field. Bradford' law, zones, and constants. The interpretation of Bradford' law in terms of a geometric progression (Y-interpretation) and its significance for calculating threshold values and generating dictionaries [**16**].

***Laboratory assignment* No 10.**

Aim: *Application of Y-interpretation of Bradford's law for creation of a reference dictionary.*
Source data: *A text*
Software: *AntConc*

1) Open the text in *AntConc* program. With the help of *Wordlist* function get raw counts for the words in the text and copy the ranked list into *Excel*.
2) Sum up all the frequencies. You will get S_n value in the formula

$$S_n = J_2(q^n - 1)/(q - 1)$$

where q is Bradford's constant=3
3) Turn the formula into the equation in one unknown and find the value of J_2 This value will be equal the sum of word counts in the lowest part of the ranked list, i.e. this is numerical value of J_2 Bradford zone.
4) Multiply the J_2 value by the Bradford constant to get the numerical value of J_1. Multiply the numerical value of J_1 zone by the Bradford constant to get the numerical value of J_0 zone. You have the numerical values for all three zones.
5) Find how many words each of three zones comprises. Detect threshold values empirically selecting randomly the last element of the zone and modifying the range of the zone. Highlight the boundaries between zones with color. Calculate the number of words in each zone.
6) Define semantic and statistical differences between the three zones.

Lesson 8.

Zipf's law; its three constants [**17**]. The notion of a rank and dependency between the rank and object's frequencies. Predictive strength of the law. Deviations from the law as indicators of crisis development.
Hartley-Shannon law; entropy and information; additivity and monotonicity [**18**]. Differences between Hartley formula and Shannon formula; parametric indicators; equally likely events. How to compute information quantity using Shannon formula; differences between treatment of information in computer science and cybernetics.

***Laboratory assignment* No 11.**

Aim: *Application of Zipf's law for analysis of text data.*
Source data: *Two texts*
Software: *AntConc*

1) Get source data for computations according to Zipf's law. With *AntConc* get raw word counts from two texts. Save results first in a .txt file and then in Excel as a ranked list. For each word calculate its probability value.
2. For each text create Zipf's distribution of words' probability values and compare this distribution with the real one. Define absolute differences between Zipf's distribution and the real one.
For example, the probability value of the word with the first rank = 0,53, while that one of the word with the second rank = 0,34. Hence the real distribution deviates from the Zipf's distribution, which should be: 0,53/2 =0,265. The deviation degree will be equal to the absolute difference between the real value 0,34 and Zipf's value 0,265.
3) Create diagrams representing the two distributions for the first 200 words.

4). Define mean deviations from Zipf's distribution in both texts. Sum up differences between the two distributions and divide the result by the total number of tokens.
In which text is the deviation from Zipf's law greater? Why?

***Laboratory assignment* No 12.**

Aim: *Application of Shannon formula for analysis of text data.*
Source data: *Table with probability values for symbols of Russian and English alphabets, two texts.*

1) Using Shannon formula and data from Table 2 calculate the quantity of information for one symbol of Russian alphabet. Compare the obtained value with that one computed according to Hartley formula. Explain the difference.

Table 2. Probability values for symbols of Russian alphabet.

i	Symbol	P(i)	I	Symbol	P(i)	i	Symbol	P(i)
1	_	0.175	12	Л	0.035	23	Б	0.014
2	О	0.090	13	К	0.028	24	Г	0.012
3	Е	0.072	14	М	0.026	25	Ч	0.012
4	Ё	0.072	15	Д	0.025	26	Й	0.010
5	А	0.062	16	П	0.023	27	Х	0.009
6	И	0.062	17	У	0.021	28	Ж	0.007
7	Т	0.053	18	Я	0.018	29	Ю	0.006
8	Н	0.053	19	Ы	0.016	30	Ш	0.006
9	С	0.045	20	З	0.016	31	Ц	0.004
10	Р	0.040	21	Ь	0.014	32	Щ	0.003
11	В	0.038	22	Ъ	0.014	33	Э	0.003
						34	Ф	0.002

2. Using Shannon formula, calculate the quantity of information in the English text and its Russian equivalent (below). For each symbol define its probability value and create the table analogous to Table 2.

This article provides extensive evidence that in German the oblique cases dative and genitive require overt morphological case marking whereas the structural cases nominative and accusative may be assigned independently. This is demonstrated in the following tests: 1. Function changing operations; 2. non-inflecting nominals; 3. binding; 4. secondary predication; 5. extraction; 6. topic drop.

В статье приводятся убедительные доказательства того, что в немецком языке косвенные падежи, такие как дательный и родительный, требуют эксплицитной морфологической маркировки, в то время как структурные падежи, такие как именительный и винительный, могут не зависеть от морфологических характеристик слова. Этот вывод подтверждается следующими тестами. 1. Процедурой изменения функций; 2. нефлективными номинативными группами; 3. связыванием; 4. вторичной предикацией; 5. экстрагированием; 6. опущением темы.

3) Calculate the total quantity of information in each of the texts. Which text contains more information? Why?

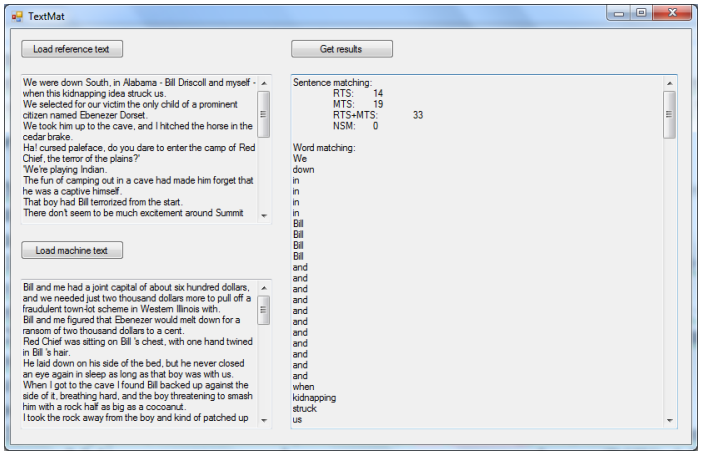
Lesson 8.

Evaluation of linguistic software. Intrinsic and extrinsic methods. TIPSTER SUMMAC project for summarization assessment; matching of a machine text against a reference text, DUC project, A. Nenkova and R. Passonneau. Semantic and lexical methods for summarization evaluation; identification of propositions and key words [**19**]. Matching against the source text and against the reference summary. The conception of a reference dictionary and methodologies for its creation and generation.
Evaluation of information retrieval systems. Types of queries and information retrieval systems; matching of query terms against the results of search. The conception of depth of search [**20**].

***Laboratory assignment* No 13.**

Aim: *Evaluation of summarizers by matching their output against a reference summary*.
Source data: *Machine summaries, reference summaries*
Software: *Subject Search Summarizer* (*SSS*), *Copernic Summarizer*, *Text matching program* (*TextMat*) (developed at the CLL), *AntConc*.
Load to *TextMat* the summaries obtained from *Subject Search Summarizer* (*SSS*) and *Copernic Summarizer*.

1) Load to *TextMat* the reference summary and summaries generate by *SSS* and *Copernic* summarizers. You will get sentences and words identical in the reference and machine summaries as well as the following statistical data.
RTS – number of sentences in the reference summary.
MTS – number of sentences in the machine summary.
NSM – number of identical sentences in the reference summary and the machine summary.
RSW – number of words in the reference summary.
MSW - number of words in machine summary.
NWM - number of identical words in the reference summary and the machine summary.
Number of identical unigrams and digrams.



For each summary calculate scores described below.
2) Divide the number of identical sentences by the total number of sentences in machine text and reference text to get the NSM numerical value. To this numerical value add additional score ANSM = NSM/2.
3) Copy identical words to a .txt file and use *AntConc* to get their frequencies. Save the results in a .txt file and then copy them to Excel. Calculate for each word a probabilistic value. Create ranked list of words.
4) Follow the methodology described in *Laboratory assignment 10* to divide the ranked list into 3 zones – J_0, J_1, J_2. Identify the words that constitute J_1 and sum up their numerical values to get J_1 score for each summary.
5) Add to J_1 score the *ANSM* score computed earlier. The sum of the scores will be the resulting score for each summary.
Compare summaries' scores. The summary with the bigger score will have a better quality.

REFERENCES

1. Yatsko V. A. Methods and Algorithms for Automatic Text Analysis In: Automatic Documentation and Mathematical Linguistics. 2011. Vol. 45. No. 5. P. 224–231..
2. What is stemming? http://www.comp.lancs.ac.uk/computing/research/stemming/general/
3. Stemming errors http://www.comp.lancs.ac.uk/computing/research/stemming/general/stemmingerrors.htm
4. Kilgarriff, Adam. BNC database and word frequency lists http://www.kilgarriff.co.uk/bnc-readme.html
5. Yatsko V. A., Starikov M. S., Larchenko E. V. et al. The algorithms for preliminary text processing: Decomposition, annotation, morphological analysis. In: Automatic Documentation and Mathematical Linguistics. 2009. Volume 43. Number 6. P.336-343
6. Brill, Eric. A simple part-of-speech tagger http://acl.ldc.upenn.edu/H/H92/H92-1022.pdf
7. Mustafaraj E., Hoof V., Freisleben D. Mining diagnostic text reports by learning to annotate knowledge roles In: Natural language processing and text mining / Ed-s Kao A., Poteet S. - London, 2007. - P. 45-68.
8. Zhang S., Dong N. An effective combination of different order n-grams http://aclweb.org/anthology/Y/Y03/Y03-1028.pdf
9. Brinton L.J. The structure of modern English. – Amsterdam; Philadelphia: John Benjamins, 2000. – 335 p.
10. The Stanford Parser: A statistical parser http://nlp.stanford.edu/software/lex-parser.shtml
11. Yatsko V. A. Problems and algorithms of text decomposition by clauses In: Automatic documentation and mathematical linguistics. 2012. Vol. 46. No.3. P.146–152.
12. Rhetorical structure theory. http://www.sfu.ca/rst/01intro/intro.html
13. Lappin S., Leass H.J. An algorithm for pronominal anaphora resolution // Computational linguistics. 1994. - V.20. - № 4. - P.535-561. http://acl.ldc.upenn.edu/J/J94/J94-4002.pdf
14. What is WordNet? http://wordnet.princeton.edu/
15. Yatsko V. A., Starikov M. S. On the experience of designing an ontology for automatic analysis of user sentiments about commercial products. In: Automatic documentation and mathematical linguistics. 2011. Vol. 45. No. 4. P. 163–168.
16. Yatsko V. A. The interpretation of Bradford's law in terms of geometric progression In: Automatic documentation and mathematical linguistics. 2012. Vol. 46. No. 2.P. 112–117.
17. Zipf's Law http://language.worldofcomputing.net/nlp-overview/zipfs-law.html
18. Барвенов С.А. Предмет и основные понятия информатики http://www.law.bsu.by/pub/11/barvenov_5.pdf
19. Nenkova A., Passonneau R. Evaluating content selection in summarization: the pyramid method. – New York: Columbia University, 2005. – 8 p. http://www1.cs.columbia.edu/~ani/papers/pyramid.pdf
20. Козлов М.В. Яцко В.А. Метод оценки эффективности функционирования современных информационно-поисковых систем Интернета // Компьютерная лингвистика и интеллектуальные технологии – М., 2006. – С.259-264. http://www.dialog-21.ru/digests/dialog2006/materials/html/Kozlov.htm