Some Thoughts on Using Annotated Suffix Trees for Natural Language Processing

Ekaterina Chernyak

National Research University – Higher School of Economics
Moscow, Russia
echernyak@hse.ru

Abstract. The paper defines an annotated suffix tree (AST) - a data structure used to calculate and store the frequencies of all the fragments of the given string or a collection of strings. The AST is associated with a string to text scoring, which takes all fuzzy matches into account. We show how the AST and the AST scoring can be used for Natural Language Processing tasks.
Keywords: text representation, annotated suffix tree, text summarization, text categorization

1 Introduction

Natural Language Processing tasks require a text being represented by a sort of a formal structure to be processed by a computer. The most popular text representation is the Vector Space Model (VSM), designed by Salton [1]. The idea of the VSM is simple: given a collection of texts, represent every text as a vector in a space of terms. A term is a word itself or a lemmatized word or the stem of a word or any other meaningful part of the word. The VSM is widely used in any kind of Natural Language Processing tasks. The few exceptions are machine translation or text generation, when word order is important, while the VSM completely loses it. For these purposes Ponte and Croft introduced the language model [2], which is based on calculating the probability of the sequence of n words or characters, so-called n-grams. There is one more approach to text representation, which is based on suffix trees and suffix arrays. Originally the suffix tree was developed for fuzzy string matching and indexing [3]. However there appear to be several application of suffix trees to Natural Language Processing. One of them is document clustering, presented in [4]. When some sort of probability estimators of the paths in the suffix tree are introduced, it can be used as a language model for machine translation [5] and information retrieval [6].

In this paper we are going to concentrate on the so-called annotated suffix tree (AST), introduced in [8]. We will present the data structure itself and several Natural Language Processing tasks where the AST representation is successfully used. We are not going to make any comparisons to other text representation models, but will show that using the AST approach helps to overcome some exciting problems. The paper is organized as follows: the Section 2 presents the definition of the AST and the algorithm for the AST construction, Sections from 3 to 7 present exciting applications of the AST (almost all developed with author's contribution), Section 8 lists some future application, Section 9 suggests how to compare the AST scoring to other approaches, Section 10 is devoted to the AST scoring implementation. Section 11 concludes.

The project is being developed by the "Methods of web corpus collection, analysis and visualisation" research and study group under guidance of prof. B. Mirkin (grant 15 - 05 - 0041 of Academic Fund Program).

2 Annotated suffix tree

2.1 Definition

The suffix tree is a data structure used for storing of and searching for strings of characters and their fragments [3]. When the suffix tree representation is used, the text is considered as a set of strings, where a string may be any significant part of the text, like a word, a word or character n-gram or even a whole sentence. An annotated suffix tree (AST) is a suffix tree whose nodes (not edges!) are annotated by the frequencies of the strings fragments.

An annotated suffix tree (see Figure 1)[7] is a data structure used for computing and storing all fragments of the text and their frequencies. It is a rooted tree in which:

- Every node corresponds to one character
- Every node is labeled by the frequency of the text fragment encoded by the path from the root to the node.

2.2 AST construction

Our algorithm for constructing an AST is a modification of the well-known algorithm for constructing suffix trees [3]. The algorithm is based on finding suffixes and prefixes of a string. Formally, the i-th suffix of the sting is the substring, which starts at i-th character of the string. The i-th prefix of the string is the substring, that ends on the i-th character of the string. The AST is built in an iterative way. For each string, its suffixes are added to the AST one-by-one starting from an empty set representing the root. To add a suffix to the AST, first check, whether there is already a match, that is, a path in the AST that encodes / reads the whole suffix or its prefix. If such a match exists, we add 1 to all the frequencies in the match and append new nodes with frequencies 1 to the last node in the match, if it does not cover the whole suffix. If there is no match, we create a new chain of nodes in the AST from the root with the frequencies 1.

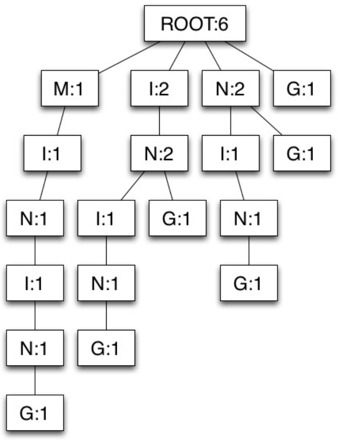

Fig. 1. An AST for string "mining".

2.3 AST relevance measure

To use an AST to score the string to text relevance we first build an AST for a text. Next we match the string to the AST to estimate the relevance.

A procedure for computing string-to-text relevance score:
Input: string and AST for a given text.
Output: the AST scoring.

1. The string is represented by the set of its suffixes;
2. Every suffix is matched to the AST starting from the root. To estimate the match we use the average conditional probability of the next symbol:

   $$score(match(suffix, ast)) = \sum_{node \in match} \phi\left(\frac{\frac{f(node)}{f(parent(node))}}{|suffix|}\right) ,$$

   where $f(node)$ is the frequency of the matching node, $f(parent(node))$ is it's parent frequency, and $|suffix|$ is the length of the suffix;
3. The relevance of the string is evaluated by averaging the scores of all suffixes:

   $$relevance(string, text) = SCORE(string, ast) = \\ = \frac{\sum_{suffix} score(match(suffix, ast))}{|string|},$$

   where $|string|$ is the length of the string.

Note, that "score" is found by applying a scaling function to convert a match score into the relevance evaluation. There are three useful scaling functions, according to experiments in [8] for spam classification:

- Identity function: $\phi(x) = x$
- Logit function:

$$\phi(x) = \log \frac{x}{1-x} = \log x - \log(1-x)$$

- Root function $\phi(x) = \sqrt{x}$

The identity scaling stands for the conditional probability of characters averaged over matching fragments (CPAMF).

Consider an example to illustrate the described method. Let us construct an the for the string “mining”. This string has six suffixes: “mining”, “ining”, “ning”, “ing”, “ng”, and “g’ . We start with the first suffix and add it to the empty AST as a chain of nodes with the frequencies equal to unity. To add the next suffix, we need to check whether there is any match, i.e. whether there is such a path in the AST starting at its root that encodes / reads a prefix of “ining”. Since there is no match between existing nodes and the second suffix, we add it to the root as a chain of nodes with the frequencies equal to unity. We repeat this step until a match is found: a prefix of the fourth suffix “ing” matches the second suffix “ining”: two first letters, “in”, coincide. Hence we add 1 to the frequency of each of these nodes and add a new child node “g” to the leaf node “n” (see Figure 1). The next suffix “ng” matches the third suffix and we repeat the same actions: increase the frequency of the matched nodes and add a new child node that does not match. The last suffix does not match any path in the AST, so again we add it to the AST’s root as a single node with its frequency equal to unity. Now let us calculate the relevance score for string “dining” using the AST in Figure 1. There are six suffixes of the string “dining”: ‘dining”, “ining”, “ning”, “ing”, “ng”, and “g’ . Each of them is aligned with an AST path starting from the root. The scorings of the suffixes are presented in Table 1.

Table 1. Computing the string “dining” score

Suffix	Match	Score
“dining”	None	0
“ining”	“ining”	$\frac{1/1+1/1+1/2+2/2+2/6}{5} = 0.76$
“ning”	“ning”	$\frac{1/1+1/1+1/2+2/6}{4} = 0.71$
“ing”	“ing”	$\frac{1/2+2/2+2/6}{3} = 0.61$
“ng”	“ng”	$\frac{1/2+2/6}{2} = 0.41$
“g”	“g”	$\frac{1/6}{1} = 0.16$

We have used the identity scaling function to score all 6 suffixes of the string “dining”. Now, to get the final CPAMF relevance value we sum and average them:

$$relevance(dining, mining) = \frac{0 + 0.76 + 0.71 + 0.61 + 0.41 + 0.16}{6} = \\ = \frac{2.65}{6} = 0.44$$

In spite of the fact that "dining" differs from "mining" by just one character, the total score, 0.44, is less than unity. This is not only because the trivial suffix "dining" contributes 0 to the sum, but also because conditional probabilities get smaller for the shorter suffixes.

3 Spam filtering

The definition of the AST presented above was for first time introduced by Pampapathi, Mirkin and Levene in [7] for spam filtering. The AST was used as a representation tool for every class (spam and ham). By introducing a procedure for scoring the class AST they developed a classifier that beats the Naive Bayes classifier in a series of experiments on standard datasets. The success of ASTs in domain of email filtering was due to the notion of match permutation normalization, which allowed to take into account some intentional typos developed by spamers to pass over spam filters. Match permutation normalization is in a a sense analogous to the edit distance [10] that if frequently implemented in spam filters [11].

4 Research paper categorization

The problem of text categorization is formulated as follows. Given a collection of documents and a domain taxonomy, annotate a document with relevant taxonomy topics. A taxonomy is a rooted tree, such that every node corresponds to a (taxonomy) topic of the domain. The taxonomy generalizes the relation "is – a" or "is a part of".

There are two basic approaches to the problem of text categorization: supervised and unsupervised. Supervised approaches give high precision values when applied to web document categorization [12], but may fail when applied to research paper categorization, since the research taxonomies, such as ACM Computing Classification System [13], are seldom revised and the supervised techniques may overfit [14]. The unsupervised approaches to text categorization are based on information retrieval – like idea: given the set of taxonomy topics, let us find those research papers that are relevant to every topic. The question for researcher is the following: what kind of the relevance model and measure to choose? In [15] we experimentally compared cosine relevance function, which measures the cosine between $tf - idf$ vectors in Vector Space Model [1], BM25, based on the probabilistic relevance framework, and the AST scoring, introduced above. These three relevance measures where applied to a relatively small dataset of 244 articles, published in ACM journals and the current version of ACM Computing Classification System. The AST scoring outperforms cosine and BM25 measures, by being more robust and taking not crisp but fuzzy measures into account. The next step in this research direction would be testing the AST scoring versus w-shingling procedure [17], which is also a fuzzy matching technique that requires text preprocessing, such stemming or lemmatization. However there is no need in stemming or lemmatization to apply the AST scoring.

5 Taxonomy refinement

Taxonomies are widely used to represent, maintain and store domain knowledge, see, for example SNOMED [18] or ACM CCS [13]. Domain taxonomy construction is a difficult task and a number of researchers have come out with idea of taxonomy refinement. The idea of taxonomy refinement is the following: having one taxonomy or upper levers of taxonomy refine it with topics extracted from additional sources such as other taxonomies, web search or Wikipedia. We followed this strategy and developed a two-step approach to taxonomy refinement, presented in more details in [21]. We concentrated on taxonomies of probability theory and mathematical statistics (PTMS) and numerical mathematics (NM), both in Russian. On a first step an expert sets manually the upper layers of taxonomy. On the second step these upper layers are refined by Wikipedia category tree and the articles, belonging to this tree, from the same domain. In this study the AST scoring is used several times:

- To clear the Wikipedia data from noise;
- To assign the remaining Wikipedia categories to the taxonomy topics;
- To form the intermediate layers of the taxonomy by using Wikipedia subcategories;
- To use Wikipedia articles in each of the added category nodes as its leaves.

The Wikipedia data is rather noisy: there some articles that are stubs or irrelevant to parental categories (the categories, they belong to) and the more so there are subcategories (of a category) that are irrelevant to the parental categories. For example, we found the article “ROC curve” be irrelevant to the category “Regression analysis” and the category “Accidentally killed” to the category “Randomness”. To define what article is irrelevant we exploit the AST scoring twice:

- We scored the title of the article to the text of the article to detect stubs;
- We scored the title of the parental category to the text of the article to detect irrelevant category.

If the value of the scoring function is less than a threshold we decided that the article is irrelevant. Usually we set the threshold at 0.2. To assign the remaining Wikipedia categories to the taxonomy topics we score the taxonomy topics to all the articles in the category merged into one text. Next we found the maximum value of the scoring function and assigned the category to the corresponding taxonomy topic. Finally, we score the title of parental categories to the articles of the subcategories, merged into one. If the subcategory to category scoring is higher than the subcategory to taxonomy topic, the subcategory remains on the intermediate layer of the refined taxonomy tree under its parental category. Finally, the articles left after clearing from noise became leaves in the refined taxonomy tree. The quality of achieved PTMS and NM taxonomies is difficult to evaluate computationally, so the design of the user study is an open question.

6 Text summarization

Automatic text summarisation is one of the key tasks in natural language processing. There are two main approaches to text summarisation, called abstractive and extractive approaches [22].

According to the abstractive approach, the summary of a text is another text, but much shorter, generated automatically to make the semantic representation of the text. According to extractive approach, the summary of a text is nothing else, but some important parts of the given text, such as a set of important sentences.

The extractive summarisation problem can be formulated in the following way. Given a text T that is a sequence of sentences S that consists of words V, select a subset of the sentences S^* that are important in T. Therefore we need to define:

– what importance of a sentence is;
– how to measure importance of the sentence; Hence we need to introduce a function, $importance(s)$, which measures the importance of a sentence. The higher *importance* is, the better. Next step is to build the summary. Let us rank all the sentences according the values of *importance*. Suppose we look for the summary that consists of five sentence. Hence we take the five sentences with the highest values of *importance* and call them top-5 sentences according to *importance*. Generally, the summary of the text are the top-N sentences according to *importance* and N is set manually.

The best results for this statement of the problem are achieved by Mihalcea and Tarau [23], where $importance(s)$ is introduced as PageRank type function [24] without any kind of additional grammar, syntax or semantic information. The main idea of the suggested TextRank algorithm is to represent a text as a directed graph, where nodes stand for sentences and edges connect sequential sentences. The edges are weighted with sentence similarity. When PageRank is applied to this graph, every node receives its rank that is to be interpreted as the importance of the sentence, so that $importance(s) = PageRank(s_{node})$, where s_{node} is the node corresponding to sentence s.

To measure similarity of the sentences the authors of TextRank algorithm suggest to use the basic VSM (Vector Space Model) scheme. First every sentence is represented as a vector in space of words or stems. Next cosine similarity between those vectors is computed. We can use the AST scoring as well for scoring the similarity between two sentences. To do this we have to introduce the common tree technique.

6.1 Constructing common subtree for two ASTs

To estimate the similarity between two sentences we find the common subtree of the corresponding ASTs. We do the depth-first search for the common chains of nodes that start from the root of the both ASTs. After the common subtree is constructed we need to annotate and score it. We annotate every node of the common subtree with the averaged frequency of the corresponding nodes in initial ASTs. Consider for example two ASTs for strings "mining" and "dinner" (see Fig. 1 and Fig. 2, correspondingly). There are two common chains: "I N" and "N", the first one consists of two nodes, the second one consists of a single node. Both this chains form the common subtree. Let us annotate it. The frequency of the node "I" is equal to 2 in the first AST and to 1 in the second. Hence, the frequency of this node in the common subtree equals to $\frac{2+1}{2} = 1.5$. In the same way we annotate the node "N" that follows after the node "I" with $\frac{2+1}{2} = 1.5$ and the node "N" on the first level with $\frac{2+2}{2} = 2$. The root is annotated with the sum of the frequencies of the first level nodes that is $1.5 + 2 = 3.5$.

6.2 Scoring common subtree

The score of the subtree is the sum of scores of every chain of nodes. The score of the path is the averaged sum of the conditional probabilities of the nodes, where conditional probability of the node is the frequency of the node divided by the frequency of its parent. For example, the conditional probability of the node "G:1" on the third level of the AST on Fig. 1 is 1/2. Let us continue with the example of "mining" and "dinner". There are two chains in their common subtree: "I N" and "N". The score of "I N" chain is $(1.5/1.5 + 1.5/3.5)/2 = 0.71$, since there are 2 nodes in the chain. The score of one node chain "N" is $1.5/3.5 = 0.42$. The score of the whole subtree is $(0.71 + 0.42) = 1.13$.

The collection for experiments was made of 400 articles from Russian news portal called Gazeta.ru. The articles were marked up in a special way, so that some of sentences were highlighted because of being more important. This highlighting was done either by the author of the article or by the editor on the basis of their own ideas. In our experiments we considered those sentences as the summary of the article. We tried to reproduce these summaries using TextRank with cosine similarity measure and AST scoring.

Using this algorithm allowed us to gain around 0.05 points of precision according to cosine baseline on our own collection of Russian newspaper texts. This is a great figure for Natural Language Processing task, taking into account that the baseline precision of the cosine measure was very low. The fact that the precision is so low can be explained by some lack of consistency in the constructed collection: the authors of the articles use different strategies to highlight the important sentences. The text collection is heterogeneous: in some articles

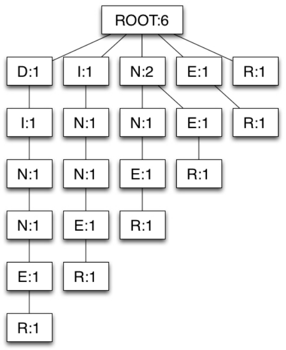

Fig. 2. An AST for string "dinner".

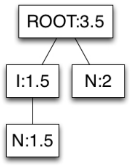

Fig. 3. Common subtree of ASTs for stings "mining" and "dinner".

there are 10 or more sentences highlighted, in some only the first one. More details of this experiment are presented in [25].

7 Association rule extraction

Several research group develop different approaches to extraction and visualization of association rules from text collections [26, 27]. Association rule is a rule $X \implies Y$, where both X and Y are sets of concepts, possibly a singleton, and the implication means some sort of co-occurrence relation. An association rule has two important features, called support and confidence. When the rule is extracted from the text collection, the support of the set X $support(X)$ usually stands for the proportion of the documents where concepts X occur and the confidence of the association rule $confidence(X \implies Y)$ stands for conditional probability of Y given X. The majority of approaches to association rule extraction share the following idea in common: the concepts should be extracted from the text collection. Using the fuzzy AST scoring we can diminish this limitation and produce the rules on the set of concepts provided by a user. In [28] we presented a so-called "conceptual map", which is a graph of association rules $X \implies Y$. To make the visualization easy we restricted ourselves only to single item sets, so that $|X| = |Y| = 1$. We analyzed a collection of Russian language newspaper articles on business and the concepts were provided by a domain expert. We used the AST scoring to score every concept k_i to every text from the collection. Next we formed $F(k_i)$ the set of articles, to which the concept k_i relevant (i.e. the scoring is higher than a threshold, usually 0.2). Finally, there was a rule $k_i \implies k_j$ if the ratio $\frac{F(k_i) \cap F(k_j)}{F(k_i)}$ was higher than the predefined confidence threshold. An example of conceptual map (translated into English) can be found on Fig. 4.

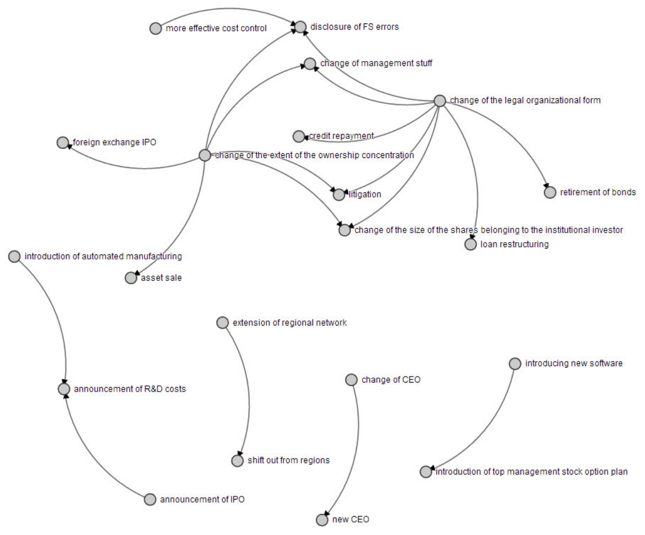
more effective cost control
disclosure of FS errors
change of management stuff
change of the legal organizational form
credit repayment
foreign exchange IPO
change of the extent of the ownership concentration
litigation
retirement of bonds
change of the size of the shares belonging to the institutional investor
loan restructuring
introduction of automated manufacturing
asset sale
extension of regional network
introducing new software
change of CEO
announcement of R&D costs
introduction of top management stock option plan
shift out from regions
announcement of IPO
new CEO

Fig. 4. A conceptual map

This conceptual map may serve as a tool for text analysis: it reveals some hidden relations between concepts and it can be easy visualized as a graph. Of course, to estimate the power of conceptual maps we have to conduct an user study.

8 Future work

In the following sections we will briefly present some Natural Language Processing tasks, where AST scoring might be used.

8.1 Plagiarism detection

Ordinary suffix trees are widely used for plagiarism detection [29]. The common subtree technique can also be used in this case. Suppose we have two texts, construct two individual ASTs and the common AST. The size of the common AST will show how much these texts share in come. Scoring the common AST allows to measure how significant coinciding parts are. With no doubts, the common AST can be used for indexing of coinciding parts of the texts. Hence, it inherits advantages of ordinary suffix trees with some additional functionality.

8.2 Compound splitting

Splitting compounds, such as German compounds, is necessary for machine translation and information retrieval. The splitting is usually conducted according to some morphological or probabilistic models [30]. We have a hypothesis that scoring prefixes of compound words to the AST, constructed from the collection of simple words, will allow to split compounds without using additional morphological knowledge. The main research in this direction is the design of the collection of simple words.

8.3 Profanity filtering

The Russian profanity language is rich and complex and has a complex derivation, usually based on adding prefixes (such as "za", "pro", "vy", etc). New words appear almost every month, so it is difficult to maintain a profanity dictionary. Profanity filtering is an important part of Russian Text or Web mining, specially since some special limitations on using profanity were introduced. The task is to find words in a text that are profane and, for example, to replace them with star symbols "***". Note, that Russian derivative includes also a variety of endings, so lematization or stemming should be used. Since Porter stemmer [31] does not cope with prefixes, it can be easily replaced by some sort of the AST-scoring.

9 Comparison to other approaches

Cosine measure on $tfidf$ vectors is a traditional baseline in majority of Natural Language Processing tasks and is easily overcame by any sort of more robust and fuzzy similarity or relevance measure, such as w-shingling [17], super shingles [32], mega shingles [33] and character n-grams [34]. The main future research concentrates on drawing comparison between these fuzzy measure and AST scoring.

10 Implementation

Mikhail Dubov's implementation of AST construction and scoring is based on suffix arrays, which makes it space and time efficient. It is available at `https://github.com/msdubov/AST-text-analysis`. It can be used as console utility or as a Python library.

11 Conclusion

In this paper the notion of annotated suffix tree is defined. The annotated suffix trees are used by several research groups and in the paper several finished, running or future projects are presented. The annotated suffix tree is a simple but powerful tool for scoring different types of relevance or similarity. This paper may sound light weighted and to make it more theoretical, we will conclude by provided some insights on probabilistic or morphological origins of ASTs. From one point of view, we have a strong feeling that it can proved that the AST or the common AST is a string kernel, thus it can be used to generate features for text classification / categorization or to measure similarity. From another point of view, the AST is a sort of supervised stemmer, that can be used to generate terms more efficiently than model-based stemmers.

12 Acknowledgments

I am deeply grateful to my supervisor Dr. Boris Mirkin for an oppurtunity to learn from him and to work under his guidance for so long and to my colleagues Mikhail Dubov, Maxim Yakovlev, Vera Provotorova and Dmitry Ilvovsky for collaboration.

References

1. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Information Processing and Management, Vol.2, no 5, pp. 513-523, 1998.
2. Ponte, J. M., and Croft B.W.. A language modeling approach to information retrieval. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 275-281. ACM, 1998.
3. Gusfield D., Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
4. Zamir O., Etzioni, O. Web document clustering: A feasibility demonstration. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 46-54. ACM, 1998.
5. Kennington C.R., Kay M. , Friedrich. A.. Suffix Trees as Language Models. In LREC, pp. 446-453. 2012.
6. Huang J.H., Powers D.. Suffix tree based approach for chinese information retrieval. Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on, vol. 3, pp. 393-397. IEEE, 2008.
7. Pampapathi R., Mirkin B., Levene M., A suffix tree approach to anti-spam email filtering, Machine Learning, 2006, Vol. 65, no.1, pp. 309-338.
8. Chernyak E.L., Chugunova O.N., Mirkin B.G., Annotated suffix tree method for measuring degree of string to text belongingness, Business Informatics, 2012. Vol. 21, no.3, pp. 31-41 (in Russian).
9. Chernyak E.L., Chugunova O.N., Askarova J.A., Nascimento S., Mirkin B.G., Abstracting concepts from text documents by using an ontology, in Proceedings of the 1st International Workshop on Concept Discovery in Unstructured Data. 2011, pp. 21-31.

10. Levenshtein, V. I., Binary codes capable of correcting deletions, insertions, and reversal. Soviet Physics Doklady Vol.10, no 8, pp. 707710.
11. Tretyakov K., Machine learning techniques in spam filtering. Data Mining Problem-oriented Seminar, MTAT, vol. 3, no. 177, pp. 60-79. 2004.
12. M. Ceci and D. Malerba Classifying web documents in a hierarchy of categories: a comprehensive study. Journal of Intelligent Information Systems, Vol. 28, no. 1, pp. 37-78, 2007.
13. ACM Computing Classification System (ACM CCS), 1998, available at: http://www.acm.org/about/class/ccs98-html
14. A.P. Santos and F. Rodrigues. Multi-label hierarchical text vlassification using the ACM taxonomy Proceedings of 14th Portuguese Conference on Artificial Intelligence, pages 553 - 564, Aveiro, Portugal, 2010.
15. Chernyak E. L. An approach to the problem of annotation of research publications, Proceedings of The Eighth International Conference on Web Search and Data Mining, pp. 429-434.
16. S. Robertson and H. Zaragoza. The probabilistic relevance gramework: BM25 and beyond. Journal Foundations and Trends in Information Retrieval, Vol.25, no 4., pp. 333-389, 2009
17. Manber, Udi. Finding Similar Files in a Large File System. Usenix Winter, vol. 94, pp. 1-10. 1994.
18. SNOMED CT - Systematized Nomenclature of Medicine Clinincal Terms, www.ihtsdo.org/snomed-ct/, visited 09.25.14.
19. Van Hage W.R., Katrenko S., Schreiber G., A Method to Combine Linguistic Ontology-Mapping Techniques, in Proceedings of 4th International Semantic Web Conference, 2005, pp. 34-39.
20. Grau B.C., Parsia B., Sirin E. Working with Multiple Ontologies on the Semantic Web, in Proceedings of the 3d International Semantic Web Conference, 2004, pp. 620-634.
21. Chernyak E. L., Mirkin B. G. Refining a Taxonomy by Using Annotated Suffix Trees and Wikipedia Resources. Annals of Data Science. Vol. 2. No. 1. P. 61-82, 2015.
22. Hahn U., Mani I. The challenges of automatic summarization, Computer, Vol.33, no.11, pp. 29-36, 2000
23. Mihalcea R., Tarau P. TextRank: bringing order into text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 404-411, 2004
24. Brin S., Page L. The anatomy of a large-scale hypertextual Web search engine. Proceedings of the seventh international conference on World Wide Web 7, 107-117, 1998
25. , Chernyak E.L., Yakovlev M.S., Using annotated suffix tree similarity measure for text summarization (under revision)
26. Pak Chung W., Whitney P., Thomas J.. Visualizing association rules for text mining. Information Visualization, 1999. Proceedings. 1999 IEEE Symposium on, pp. 120-123. IEEE, 1999.
27. Mahgoub, H., Rsner, D., Ismail, N., Torkey, F.. A text mining technique using association rules extraction. International journal of computational intelligence 4, no. 1, pp. 21-28, 2008.
28. Morenko, E. N., Chernyak E.L., Mirkin B.G.. Conceptual Maps: Construction Over a Text Collection and Analysis. In Analysis of Images, Social Networks and Texts, pp. 163-168. Springer International Publishing, 2014.

29. Krisztin M., Zaslavsky A., Schmidt, H.. Document overlap detection system for distributed digital libraries. Proceedings of the fifth ACM conference on Digital libraries. ACM, 2000.
30. Koehn P., Knight K.. Empirical methods for compound splitting. Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003.
31. Porter, M. F. An algorithm for suffix stripping. Program Vol. 14, no. 3, pp, 130-137 (1980).
32. Chowdhury A., Frieder O., Grossman D., McCabe M.C..”Collection statistics for fast duplicate document detection. ACM Transactions on Information Systems (TOIS) Vol. 20, no. 2 ,pp. 171-191 (2002).
33. Conrad J. G., Schriber C.P.. Managing dj vu: Collection building for the identification of nonidentical duplicate documents. Journal of the American Society for Information Science and Technology Vol 57, no. 7 pp. 921-932 (2006).
34. Damashek M.. Gauging similarity with n-grams: Language-independent categorization of text. Science Vol. 267, no. 5199, pp 843-848 (1995).