Our future direction is the investigation of the machine learning algorithm applications in the *dynamic* mode. In this case the feature vectors are extracted consequently every short period of time (for example each 0,01 sec.). Moreover, speaker specific and gender specific information should be used in order to improve the emotion recognition accuracy from speech. The emotion recognition accuracy (as well as a SDS's performance in general) might be significantly improved by training of the speaker specific emotional models and using gender specific information as well. The next step is the exploitment of the best algorithms for emotion recognition and speaker identification from speech in order to build a speaker dependent emotion recognition systems.

References

1. Burkhardt F., Paeschke A., Rolfes M., Sendlmeier W., & Weiss B. (2005, September). *A database of German emotional speech*. In Proc. Interspeech (Vol. 2005).
2. Maxine Eskenazi, Alan W Black, Antoine Raux, and Brian Langner. *Let's Go Lab: a platform for evaluation of spoken dialog systems with real world use* In Proceedings of Interspeech 2008 Conference, Brisbane, Australia.
3. Alexander Schmitt, Benjamin Schatz and Wolfgang Minker. *Modeling and predicting quality in spoken human-computer interaction.* In Proceedings of the SIGDIAL 2011 Conference, Association for Computational Linguistics, 2011.
4. Schmitt A., Heinroth T. and Liscombe J. *On NoMatchs, NoInputs and BargeIns: Do Non-Acoustic Features Support Anger Detection?* Proceedings of the SIGDIAL 2009. Conference, Association for Computational Linguistics, London, UK, 2009, p. 128–131.
5. Haq S. and Jackson P. J. B. *Multimodal Emotion Recognition*. In W. Wang (ed), Machine Audition: Principles, Algorithms and Systems, IGI Global Press, ISBN 978-1615209194, chapter 17, 2010, p. 398–423.
6. Available at: http://uudb.speech-lab.org/.
7. Schuller B., Vlasenko B., Eyben F., Rigoll G., & Wendemuth A. (2009, November). *Acoustic emotion recognition: A benchmark comparison of performances.* In Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on, p. 552–557.
8. Michael Grimm, Kristian Kroschel, and Shrikanth Narayanan. *The Vera am Mittag German Audio-Visual Emotional Speech Database*. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), Hannover, Germany, 2008.
9. Available at: http://www.einslive.de/sendungen/domian/.
10. Bogdanov D. S., Bruhtiy A. V., Krivnova O. F., Podrabinovich A. Ya. and Strokin G. S., 2003. *Organizational Control and Artificial Intelligence, chapter Technology of Speech Databases Development (in Russian)*, p. 448. Editorial URSS.
11. Zablotskiy S., Shvets A., Sidorov M., Semenkin E. and Minker W. *Speech and Language Resources for LVCSR of Russian.* In Proceedings of the LREC 2012, Istanbul.
12. Available at: http://www.speech.cs.cmu.edu/databases/pda/README.html.
13. Sidorov M., Schmitt A., Zablotskiy S. and Minker W. *Survey of Automated Speaker Identification Methods.* In Proceedings of Intellegent Environment 2012, Athens, Greece. In Press.
14. Boersma Paul & Weenink David (2013). Praat: doing phonetics by computer [Computer program]. Version 5.3.50, retrieved 21 May 2013 from http://www.praat.org/.
15. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009). The WEKA Data Mining Software: An Update; SIGKDD Explorations, vol. 11, Issue 1.
16. Mierswa, Ingo and Wurst, Michael and Klinkenberg, Ralf and Scholz, Martin and Euler, Timm. *YALE: Rapid Prototyping for Complex Data Mining Tasks*. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.

УДК 519.6+004.9

TWO-STEPS SYSTEM IN SEARCHING SIMILAR WORDS FOR FAST AND RELIABLE AUTOMATIC CONCATENATION OF RUSSIAN SUB-WORD UNITS

A. Spirina[1], S. G. Zablotskiy[2], M. Yu. Sidorov[2]

[1]Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation. E-mail: s_nastia@mail.ru
[2]Ulm University
43, Albert-Einstein-Allee, Ulm, 89081, Germany.
E-mail: sergey.zablotskiy@uni-ulm.de, maxim.sidorov@uni-ulm.de

In this paper we describe and investigate the two-steps system sorting out inappropriate words in searching of similar words in the lexicon for automatic concatenation of Russian sub-word units. This two-steps system consists of com-

puting the Levenshtein distance on the first stage and computing the similarity coefficient by the relevance function on the second stage. We also compared the performance of the Wagner-Fisher algorithm and the suggested algorithm SAWT.

Keywords: fuzzy search algorithm, Levenshtein distance, relevance function.

ДВУХУРОВНЕВАЯ СИСТЕМА ПОИСКА СХОЖИХ СЛОВ ДЛЯ ЭФФЕКТИВНОЙ И НАДЕЖНОЙ АВТОМАТИЧЕСКОЙ КОНКАТЕНАЦИИ СЛОГОВ РУССКОЙ РЕЧИ

А. Спирина[1], С. Г. Заблоцкий[2], М. Ю. Сидоров[2]

[1] Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: s_nastia@mail.ru
[2]Университет города Ульма
Германия, 89081, Ульм, Аллея Альберта Эйнштейна, 43
E-mail: sergey.zablotskiy@uni-ulm.de, maxim.sidorov@uni-ulm.de

Описывается и исследуется двухуровневая система отбора наиболее подходящих слов в поиске схожих слов по словарю для автоматической конкатенации слогов русской речи. Эта система состоит из вычисления расстояния Левенштейна на первом этапе и вычисления коэффициента сходства с помощью функции релевантности на втором этапе. Мы также сравнили эффективность алгоритма Вагнера–Фишера и предлагаемого алгоритма SAWT.

Ключевые слова: алгоритм нечеткого поиска, расстояние Левенштейна, функция релевантности.

The lexicon for Russian continuous speech recognition is much larger than that for English. This fact complicates the use of standard well developed approaches to language modeling. Quite a common approach to handle an abundant lexicon is the employment of sub-units, like syllables or morphemes. The challenge of such approach is the subsequent concatenation of recognized sub-unites.

There exist some related works done to solve this problem such as [1]. However, further improvement of the concatenation accuracy and the performance of the algorithms are required.

Continuous speech is transformed into the sequence of syllables, but word boundaries are unknown. The task is to accelerate the automatic concatenation of Russian subword units. For this purpose the Genetic Algorithm (GA) can be used. However, the likelihood of the sentence generated by GA should be estimated. The problem can be partly solved by accelerating the search for the same and similar words from the lexicon to the words from GA by exploiting the fuzzy search algorithms.

Fuzzy search. Spell-checkers and different web search engines (such as Google, Yandex, etc) are also based on the fuzzy (string) search algorithms. For example, the fuzzy search algorithms are used in the web search engines to generate the results of the “Did you mean …” suggestion list [2].

The problem of the fuzzy search can be formulated as follows: “Find in the text or lexicon of size N all the words matching the original word within the maximum K possible differences” [2].

There exist different fuzzy search algorithms, such as: linear search, bitap (Shift-Or or Baeza-Yates-Gonnet, and its modifications by Wu and Manber), Signature Hashing Method and others [2].

Fuzzy search algorithms are based on some metric, i. e. distance function between two strings, which measures their similarity or difference. One of the most well-known metrics is the Levenshtein distance.

The Levenshtein distance and the Wagner-Fisher algorithm. The Levenshtein distance (edit distance) between two strings is the minimum number of single-character edits (insertion, deletion, substitution) required to transform one string into the other. [3]

Suppose S_1 and S_2 are strings, then, mathematically, the Levenshtein distance can be described by the following formula (1):

$$D_{i,j} = \begin{cases} 0, & i=0, j=0 \\ i, & i>0, j=0 \\ j, & i=0, j>0 \\ \min(D_{i,j-1}+1, D_{i-1,j}+1, D_{i-1,j-1}+C_{substitution}), & i>0, j>0 \end{cases}$$

$$C_{substitution} = \begin{cases} 1, \; if \;\; S_1[i] \neq S_2[j] \\ 0, otherwise \end{cases} \tag{1}$$

Different sources suggest $C_{substitution}$ to be equal to 2 instead of 1 in the formula (1). In the following tests $C_{substitution} = 2$ was used.

There exists a set of algorithms for computing the Levenshtein distance. Most popular algorithm is the Wagner-Fisher algorithm [4]. In this work the Wagner-Fisher algorithm was used, in which $C_{substitution}$ is presented by formula (2):

$$C_{substitution} = \begin{cases} 2 * WeightFunction(S_1[i], S_2[j]), \; if \;\; S_1[i] \neq S_2[j] \\ 0, otherwise \end{cases} \tag{2}$$

Weight Function is the function of weight coefficients for the symbol comparison. This function provides a set of rules for the phonetic comparison. It measures the phonetic similarity between two words.

To accelerate the performance the lexicon is stored in one single tree. The Levenshtein distance is computed at each node of the tree.

But the Wagner-Fisher algorithm has some drawbacks. This algorithm applied to the tree can sort out appropriate words at the beginning of the tree.

The algorithm of search appropriate words in a tree. The algorithm SAWT was developed to accelerate the search for the similar words from the lexicon and to overcome drawback of the Wagner-Fisher algorithm described above.

The ASAWT is worthy of using only if the maximum allowed distance between strings is rather small (here distance is the measure of difference), for example 2 or 3. This algorithm is able to overcome the disadvantage of the Wagner-Fisher algorithm.

The idea of the ASAWT is as follow:

First of all, suppose that:

1. The maximum allowed distance between two strings is k.
2. S_1 of size n is the original string and S_2 of size m is the test string. Furthermore, there is a restriction for n and m: $| n - m | \le k$ (this restriction saves the computational load while searching).
3. A value of the position p in the original string is known (starting from $p = 0$ corresponding to the first index in the string).
4. A value of the error err is known (starting from err = 0).

Then the mechanism of the distance computation between two strings consists of the following steps:

1. Denote $i^{\underline{th}}$ character of the string S by $S[i]$ and $j = 0$.
2. $p_s = \begin{cases} i-p, \ if \ \exists i \in [p, p+k]: S_2[j] = S_1[i]; \\ -1, \quad otherwise. \end{cases}$
3. $p = \begin{cases} p + p_s + 1, \ if \ p_s \ne -1; \\ p, \qquad otherwise. \end{cases}$
4. $err = \begin{cases} err + p_s, \ if \ p_s \ne -1; \\ err + 1, \ otherwise. \end{cases}$
5. If $err > k$, then stop the computation. It means that S_1 and S_2 are different.
6. $j = j + 1$
7. If $p < n$ AND $j < m$, then go to the step 2.
8. $err = err + n - p + m - j$.

If $err \le k$ the strings are decided to be similar, otherwise the strings are different.

For example, suppose S_1 is "TRUST" and S_2 is "TEST", $k = 1$ (left table) and $k = 3$ (right table).

It should be mentioned that the cost for insertion, deletion is 1 and for substitution is 2 like in the Wagner-Fisher algorithm whit formula (2), but without exploiting of WeightFunction.

Similarity coefficient. To get more appropriate results two-step system sorting out inappropriate words from the lexicon was applied.

At the first step the Levenshtein distance between the original word (generated by the GA) and the words from the lexicon is computed. At the second step the similarity coefficient between the original word and the word from the lexicon is computed by the Relevance function.

The similarity coefficient is the fractional number between 0 and 1, 0 means that two words are absolutely different, 1 means that two words are identical.

There exist different word similarity coefficients, such as the Sörensen coefficient, the Kulczinsky coefficient, the Ochiai coefficient, the Szymkiewicz-Simpson coefficient and Braun-Blanquet coefficient [5]. These coefficients reflect the similarity coefficient dependence on the length of the words.

For example, the formula (3) is presents the Sörensen coefficient:

$$K_S = \frac{2c}{a+b}, \tag{3}$$

where a and b are the lengths of the words, c is the number of the matching characters, which can be computed by the formula (4) where for computing the Levenshtein distance (LD) the cost for all edit operations is equal to 1.

$$c = \max(a, b) - LD\,. \tag{4}$$

The Relevance function. The Relevance function gives the similarity coefficient which allows to take into account the positions of the difference in the word. Thus the difference at the beginning or at the end of the word is less critical than in the middle of the word [6].

The similarity coefficient in this case can be computed by the formula (5) and formula (6):

$$R = \frac{\sum_{i=1}^{N} r(i)}{N}; \tag{5}$$

$$r(i) = \frac{Match(Str1, Str2, i) + Match(Str2, Str1, i)}{Count(Str1, i) + Count(Str2, i)}, \tag{6}$$

where $Match(S_1, S_2, i)$ is the number of the matches for all substrings of the length i from the string S_1 in the string S_2; $Count(Str, i) = (len(Str) - i + 1)$; $len(S)$ is the length of the string S.

For the following tests the Relevance function with $N = 2$ was used.

Certain threshold should be specified for the R to distinguish between similar and non-similar words.

Results. The computer program for testing was written in C++ using Microsoft Visual Studio 2010. The lexicon consists of 2311465 words. All tests were calculated by the computer with Intel(R) Core(TM)2 CPU 6700 @ 2.66GHz, running Linux. All results are dependent on the program realization, the characteristics of the computer, the computer workload, the words for the test and the lexicon. For testing the words of different size were randomly selected from the lexicon. The results of testing are presented in the following figures.

ASAWT for computing the distance

	T	R	U	S	T	Err
T	1					0
E		0	0			1
S		0	0			2
T						2 (err > *k*, stop)
					Distance	–

	T	R	U	S	T	Err
T	1					0
E		0	0	0	0	1
S		0	0	1		3
T					1	3
					distance	3

-

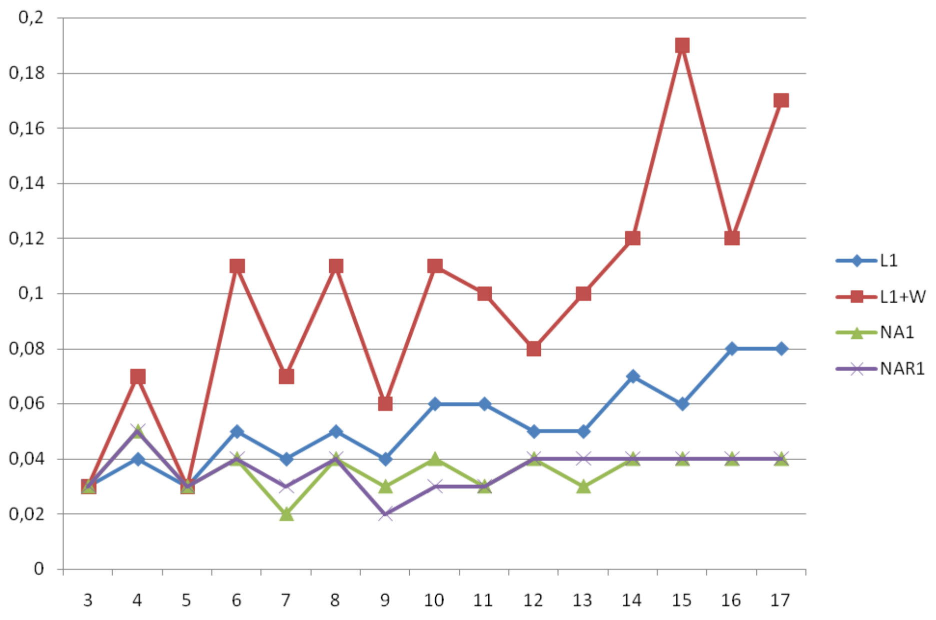

Fig. 1. Computational time for the algorithms with $k = 1$

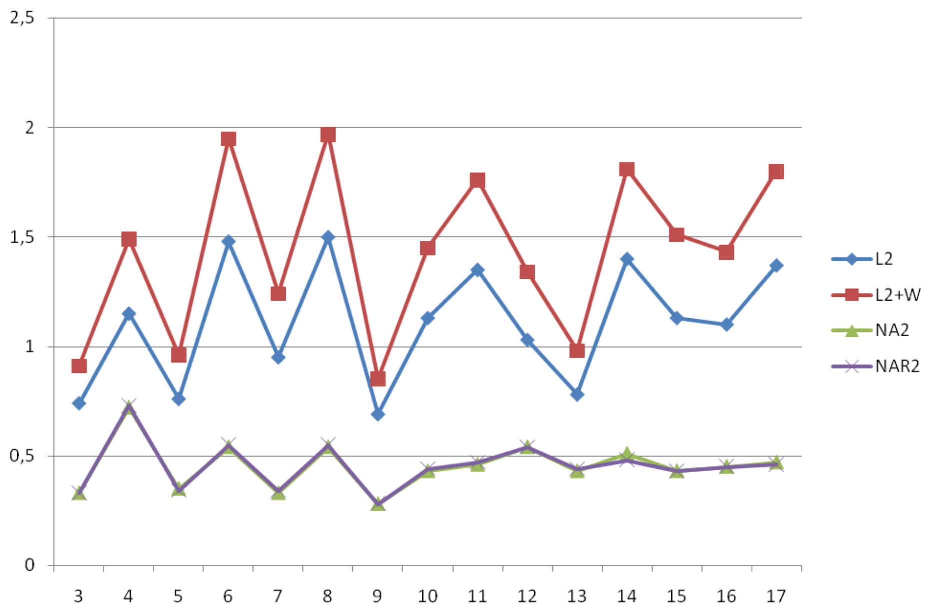

Fig. 2. Computational time for the algorithms with $k = 2$

Abbreviations on the figures:

L1, L2 mean the computation of the Levenshtein distance using the Wagner-Fisher algorithm without WeightFunction, with $k = 1$ and $k = 2$ correspondingly.

L1+W, L2+W mean the computation of the Levenshtein distance using the Wagner-Fisher algorithm with WeightFunction, with $k = 1$ and $k = 2$ correspondingly.

NA1, NA2 mean the computation of the Levenshtein distance using the ASAWT without the Relevance function, with $k = 1$ and $k = 2$ correspondingly.

NAR1, NAR2 mean the computation of the Levenshtein distance using the ASAWT with the Relevance function, with $k = 1$ and $k = 2$ correspondingly.

Computational time for 100 runs of algorithm in seconds is on the vertical axis, the length of the original words is on the horizontal axis.

Conclusion. To sum it up, from the figures above it could be seen that the ASAWT with and without relevance function works faster than the Wagner-Fisher algorithm. The final lists of similar words are almost identical. The two-step system allows to keep the accuracy at the same level.

However, the algorithm has some drawbacks, namely there are no weights modeling the closeness of the pronounced sounds. This problem is supposed to be solved in a future work.

Acknowledgements. This work is partly supported by the DAAD (German Academic Exchange Service).

References

1. Zablotskiy S., Shvets A., Sidorov M., Semenkin E. and Minker W. Speech and Language Resources for LVCSR of Russian. International Conference on Language Resources and Evaluation (LREC), Istanbul, Turkey, 2012. May.
2. Smetanin N. (2011, March 24). Fuzzy string search. *Nikita's blog. Search algorithms, software development and so on.* Retrieved July 30, 2013, from http://ntz-develop.blogspot.ru/.
3. Levenshtein V. I. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10, February.
4. Wagner R. A., Fischer M. J. The string-to-string correction problem. J. ACM. 1974, vol. 21, № 1, p. 168–173.
5. Wikipedia. Similarity index (coefficient). *Wikipedia. The free encyclopedia.* Retrieved July 30, 2013, from http://ru.wikipedia.org/ (in Russian).
6. Karakhtanov D. S. Using of fuzzy search algorithm in processing of data for credit institutions. Audit and financial analysis. 2010, vol. 2 (in Russian).

УДК 519.6

SELF-ADJUSTED EVOLUTIONARY ALGORITHMS BASED APPROACH FOR AUTOMATED DESIGN OF FUZZY LOGIC SYSTEMS

V. V. Stanovov, E. S. Semenkin

Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation
E-mail: vladimirstanovov@yandex.ru, eugenesemenkin@yandex.ru

A new approach to form a fuzzy logic system with evolutionary algorithms is introduced. Several algorithms were implemented as programs, efficiency measurements were made with and without self-adjustment. A new data representation method was developed for fuzzy rules base coding in genetic programming method by using reverse polish notation principle. The genetic algorithm was adapted to adjust linguistic variables semantics. The efficiency of developed algorithm was shown and compared to analogies on several test regression problems and real classification problems.

Keywords: genetic algorithms, genetic programming, self-adjustment, fuzzy logic system design.

САМОНАСТРАИВАЮЩИЙСЯ ЭВОЛЮЦИОННЫЙ АЛГОРИТМ ДЛЯ АВТОМАТИЗИРОВАННОГО ФОРМИРОВАНИЯ СИСТЕМ НА НЕЧЕТКОЙ ЛОГИКЕ

В. В. Становов, Е. С. Семенкин

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: nihilanht@mail.ru, eugenesemenkin@yandex.ru

Представлен новый подход для формирования систем на нечеткой логике эволюционными алгоритмами. Несколько алгоритмов были реализованы в качестве программ, проведены измерения эффективности с самонастройкой и без таковой. Был разработан новый метод представления данных для кодирования нечетких правил в генетическом программировании на основе принципа обратной польской записи. Генетический алгоритм был адаптирован для настройки семантики лингвистических переменных. Эффективность разработанного алгоритма была показана в сравнении с аналогами на некоторых тестовых регрессионных задачах и практических задачах классификации.

Ключевые слова: генетический алгоритм, генетическое программирование, самонастройка, проектирование систем на нечеткой логике.