https://doi.org/10.31891/csit-2023-2-2
UDC 004.912

Olesia BARKOVSKA, Anton HAVRASHENKO
Kharkiv National University of Radio Electronics, Ukraine

ANALYSIS OF WORD SEARCH ALGORITHMS IN THE DICTIONARIES OF MACHINE TRANSLATION SYSTEMS FOR ARTIFICIAL LANGUAGES

The main goal of the project is the analysis of word search algorithms in the dictionary for machine translation systems.
To achieve the goal, the following tasks were solved:
- analysis of shortcomings of the proposed and developed artificial language machine translation system;
- improvement of the dictionaries included in the proposed system of machine translation of artificial languages, thanks to the algorithms of searching for words of n-grams and Knuth-Morice-Pratt;
- implementation of the possibility of using the prepared dictionary for translation;
- analysis of the impact of implemented improvements to word search methods on search accuracy, choice of dictionary structure, and search time.
The paper is devoted to the development of an organizational model of the machine translation system of artificial languages. The main goal is the analysis of word search algorithms in the dictionary, which are significant elements of the developed machine translation system at the stage of improvement of new dictionaries created on the basis of already existing dictionaries. In the course of the work was developed a model of the machine translation system, created dictionaries based on texts and based on already existing dictionaries using augmentation methods such as back translation and crossover; improved dictionary based on algorithms of n-grams, Knuth-Morris-Pratt and word search in the text (such as binary search, tree search, root decomposition search). In addition, the work implements the possibility of using the prepared dictionary for translation. The obtained results can improve existing systems of machine translation of the text of artificial languages. Namely, to reduce the operating time by approximately 20 times when switching from the balanced tree algorithm to other logarithmic algorithms. The practical significance of this work is the analysis and improvement of text augmentation algorithms using algorithms of binary search, hashes, search tree, and root decomposition.

Keywords: translation, prefix tree, dictionary, artificial language, hash, binary search, search tree

Олеся БАРКОВСЬКА, Антон ГАВРАШЕНКО
Харківський національний університет радіоелектроніки

АНАЛІЗ АЛГОРИТМІВ ПОШУКУ СЛІВ ДЛЯ ЗАСТОСУВАННЯ В СИСТЕМАХ МАШИННОГО ПЕРЕКЛАДУ ШТУЧНИХ МОВ

Основною метою проекту є аналіз алгоритмів пошуку слів у словнику для систем машинного перекладу.
Для досягнення мети були вирішені наступні завдання:
- аналіз недоліків запропонованої та розробленої системи машинного перекладу штучної мови;
- удосконалення словників, що входять до складу запропонованої системи машинного перекладу штучних мов, завдяки алгоритмам пошуку слів n-грамм та Кнута-Моріса-Пратта;
- реалізація можливості використання підготовленого словника для перекладу;
- аналіз впливу впроваджених удосконалень методів пошуку слів на точність пошуку, вибір структури словника та час пошуку.
Робота присвячена розробці організаційної моделі системи машинного перекладу штучних мов.Головною метою є аналіз алгоритмів пошуку слова в словник, які є значущими елементами розробленої системи машинного перекладу на етапі вдосконалення створених нових словників на основі вже існуючих словників. В ході виконання роботи була розроблена модель системи машинного перекладу, створені словники на основі текстів та на основі вже існуючих словників методами аугментації такими, як зворотній переклад та кросовер; вдосконалено створений словник на основі алгоритмів n-грамм, Кнута-Моріса-Пратта та пошуку слів у тексті (таких, як бінарний пошук, пошук в дереві, пошук в кореневій декомпозиції). Окрім того, в роботі реалізована можливість використання підготовленого словнику для перекладу. Отримані результати можуть покращити існуючі системи машинного перекладу тексту штучних мов. А саме – зменшити час роботи приблизно у 20 разів при переході від алгоритму збалансованого дерева до інших логарифмічних алгоритмів. Практичною значущістю даної роботи є аналіз та покращення алгоритмів аугментації тексту за допомогою алгоритмів бінарного пошуку, хешів, дерева пошуку, кореневої декомпозиції.

Ключові слова: переклад, префіксне дерево, словник, штучна мова, хеш, бінарний пошук, дерево пошуку

Introduction

Machine translation (MT) – technology of automated translation of texts (written and spoken) from one natural language to another using a computer; the direction of scientific research related to the construction of automated translation systems [1,3].

At a basic level, the job of computer translation programs is to replace words or phrases from one language with words or phrases from another. However, the problem arises that such a replacement cannot provide a high-quality translation of the text, because it requires the definition and recognition of words and entire phrases from the original language. This encourages active researches in the field of computational linguistics.

Machine translation is one of the subgroups of computational linguistics that studies the use of software to translate text from one language to another [4,6]. At the initial level, MT performs the usual replacement of words from one language for words from another language, but usually the translation made in this way is not very high-

quality, because in order to fully convey the meaning of the sentence and find the closest analogue in the "target" language) – to the language required by the translator, it is often necessary to translate an entire phrase. Commonly machine translation divided on 4 methods (Figure 1).

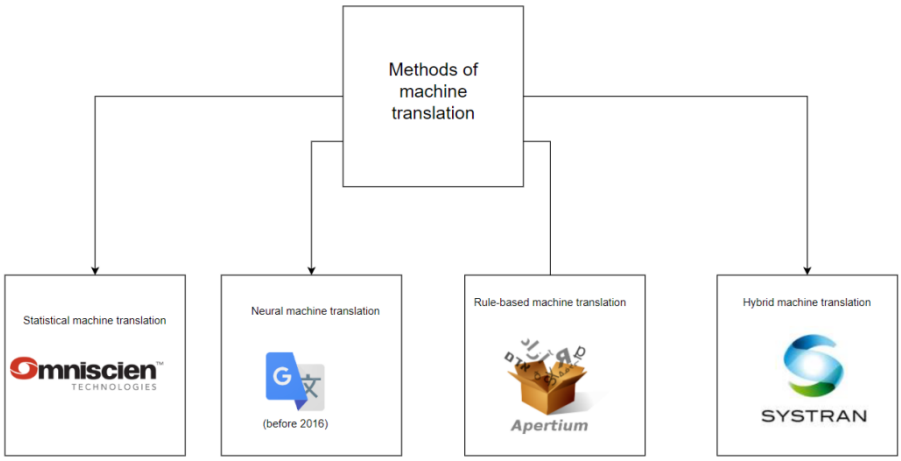


Fig. 1. Methods of machine translations

Let's take a closer look at statistical machine translation.

Statistical machine translation is a type of machine translation of text based on the comparison of large volumes of language pairs. Language pairs - texts containing sentences in one language and corresponding sentences in another can be both variants of writing two sentences by a person - a native speaker of two languages - and a set of sentences and their translations performed by a person. Thus, statistical machine translation has the property of "self-learning". The more language pairs the program has at its disposal and the more precisely they correspond to each other, the better the result of statistical machine translation [7,9].

Table 1

Comparison of translation methods

Criterion for comparison	Statistical machine translation	Neural machine translation	Rule-based machine translation	Hybrid machine translation
Translation speed	It takes the least amount of time	It takes a lot of time	It takes a lot of time	It takes the most time
Translation quality	High	The best	High	Quite high
Extensibility	There is a possibility of expansion	There is a possibility of expansion	There is a possibility of expansion, but it is more difficult than in other algorithms	There is a possibility of expansion, but it is more difficult than in other algorithms

The first ideas of statistical machine translation were presented by Warren Weaver in 1949, including the ideas of information theory applications of Claude Shannon. Statistical machine translation was reintroduced in the late 1980s and early 1990s by researchers at IBM's Thomas J. Watson Research Center and has contributed to a significant resurgence of interest in machine translation in recent years. Before the introduction of neural machine translation, it was by far the most researched method of machine translation [10,12].

In verbal translation, the basic unit of translation is a word of a certain natural language. As a rule, the number of words in translated sentences varies due to complex words, morphology and idioms. The ratio of the length of sequences of translated words is called fertility, which shows how many foreign words each native word creates. Information theory necessarily assumes that everyone embraces the same concept. In practice, this is not quite the case. For example, the English word corner can be translated into Spanish as rincón or esquina, depending on whether it means inside or outside corner.

We compare all methods according to such parameters as translation speed, quality and extensibility (Table 1).

After the comparison, we can conclude that each of the types of machine translation can be supported, but the best are neural and hybrid machine translation, which are currently used most often.

Related works

An example of a word-based translation system (Figure 2) is the freely available package GIZA++ (GPLed), which includes a tutorial for IBM models, HMM models, and Model 6 [13].

Verbal translation is not widely used today; phrasal systems are more common. Most phrase-based systems still use GIZA++ for corpus matching. Alignment is used to highlight phrases or derive syntactic rules. And matching

words in double text is still a hotly debated issue in the community. Due to the prevalence of GIZA++, there are now several distributed implementations of it online.

In phrase-based translation, the goal is to reduce the limitations of word-based translation by translating whole sequences of words where the length can vary. Sequences of words are called blocks or phrases, but usually they are not linguistic phrases, but phrases found using statistical methods from corpora. It is shown that restricting phrases to language phrases (syntactically motivated groups of words) reduces the quality of translation.[14].

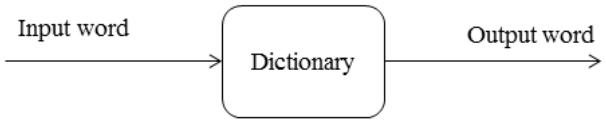


Fig.2. Diagram of the translation system based on words

The selected phrases are then displayed next to each other based on the phrase translation table and can be changed. This table can be learned from word alignment or directly from a parallel corpus.

Syntax-based translation is based on the idea of translating syntactic units rather than individual words or word strings (as in phrase-based MT), i.e. (partial) analysis of sentence/utterance trees. The idea of syntax-based translation is quite old in MT, although its statistical counterpart did not become widespread until the advent of strong stochastic parsers in the 1990s. Examples of this approach include DOP-based MT and, more recently, synchronous context-free grammars. Syntactic unit is always a combination that has at least two constituents (Figure 3). The basic syntactic units are a word-group, a clause, a sentence, and a text [15]. Their main features are:

-they are hierarchical units – the units of a lower level serve the building material for the units of a higher level;

-as all language units the syntactic units are of two-fold nature;

-they are of communicative and non-communicative nature – word-groups and clauses are of non-communicative nature while sentences and texts are of communicative nature.

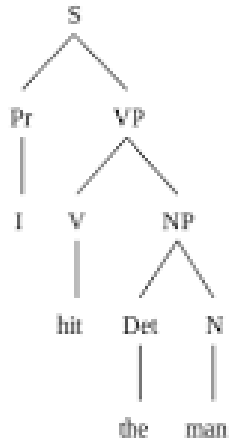


Fig.3. Example of syntactic units

Another example of a statistical machine translator is Google Translate from 2006 to 2016.

Google Translate did not apply grammar rules because its algorithms were based on statistical or pattern analysis rather than traditional rule-based analysis. The system's creator, Franz Josef Och, criticized the effectiveness of rule-based algorithms in favor of statistical approaches. The original versions of Google Translate were based on the statistical machine translation method, or rather, on the research of Och, who won the DARPA competition for high-speed machine translation in 2003. Och was the head of Google's machine translation team.

A solid foundation for developing a usable statistical machine translation system for a new language pair from scratch would consist of a bilingual text corpus (or parallel collection) of over 150-200 million words and two monolingual corpora of over a billion words each. Statistical models from this data are then used to translate between these languages.

To get such a huge amount of linguistic data, Google used documents and transcripts of the United Nations (UN) and the European Parliament. The UN usually publishes documents in all six official UN languages, creating a very large 6-language corpus.

Google representatives participated in internal conferences in Japan where they requested bilingual data from researchers (Figure 4).

Fig.4. Chronology of the organization of linguistic corpora for the operation of the Google Translate system

When Google Translate generates a translation suggestion, it looks for patterns in hundreds of millions of documents to help choose the best translation. By detecting patterns in documents that have already been translated by translators, Google Translate makes educated guesses (AI) about what the correct translation should be.

Until October 2007, for languages other than Arabic, Chinese, and Russian, Google Translate was based on SYSTRAN, a software engine still used by several other online translation services, such as Babel Fish (now defunct). Since October 2007, Google Translate has used its own technology based on statistical machine translation, and then switched to neural machine translation.

Also, Google Translate did not have the ability to translate directly from certain languages. For the translation of some languages, translations were made into English, which were then translated into the required language. For example, for a translation from Ukrainian, the text was translated into Russian, after which it was translated into English. After these translations, the obtained result was translated into the required language.

In systems that use dictionaries, searching for words is an actual problem, because depending on the selected search algorithm, it will be necessary to change the type of dictionary that will store words. For example, binary search or two pointers cannot be used on an unordered dictionary. We compare the complexity (in Big O notation) of algorithms on different operations with dictionaries (Table 2).

Table 2

Comparison of methods in Big O notation

	Binary search	**Hash table**	**Binary tree**	**Prefix tree**	**2 instructions**	**Sqrt decomposition**	**Normal search**
Creating a dictionary	*NlogN*	*N*M*	*NMlogN*	*N*M*	*NlogN*	*NlogN*	*N*
Combining two dictionaries	*N*	*P*	*NMlogN*	*M*K*	*N*	*N*	*N*
Adding a new word	*N*	*M*	*MlogN*	*M*	*N*	*N*	*1*
Word search	*logN*	*1*	*MlogN*	*M*	*N*	*Sqrt(N)*	*N*
Memory used	*N*M*	*N*M+P*	*N*M*	*M*K*	*N*M*	*(N +sqrt(N)) *M*	*N * M*

Where *N* is the number of words in the dictionary, *M* is the average word length, *P* is the hashing module .

As we can see, each of the algorithms has its advantages and disadvantages. If you need to make an application as quickly as possible, and the running time or occupied memory are not important, then it is best to use a balanced tree, because it is usually already implemented in programming languages. For example, std::map in C++ or dictionary in Python.

For fast search if there are no memory limitations, we can use hashing combined with binary search, which will give a good performance boost.

If you need to find words often, and we can afford to use a complex structure, it is recommended to use the Bohr method, which will allow you to quickly add words and quickly find them in a time close to the length of a word. But when using a large alphabet, the method loses its power.

Main goal and tasks of the research

The main goal of the project is the analysis of word search algorithms in the dictionary for machine translation systems.

To achieve the goal, the following tasks must be solved:

- analysis of shortcomings of the proposed and developed artificial language machine translation system [16];
- improvement of the dictionaries included in the proposed system of machine translation of artificial languages, thanks to the algorithms of searching for words of n-grams and Knuth-Morice-Pratt[17];
- implementation of the possibility of using the prepared dictionary for translation [18];
- analysis of the impact of implemented improvements to word search methods on search accuracy, choice of dictionary structure, and search time.

Experiments

Detailed model of the proposed machine translation system , which is the basis for conducting research and analyzing the impact of data volume on runtime, was proposed in [19].

The system[20] model includes the following modules: a module for generating dictionaries based on input texts, a module for generating dictionaries based on dictionaries of other languages; and a translator program for translating texts with the help of dictionaries(Figure 5).

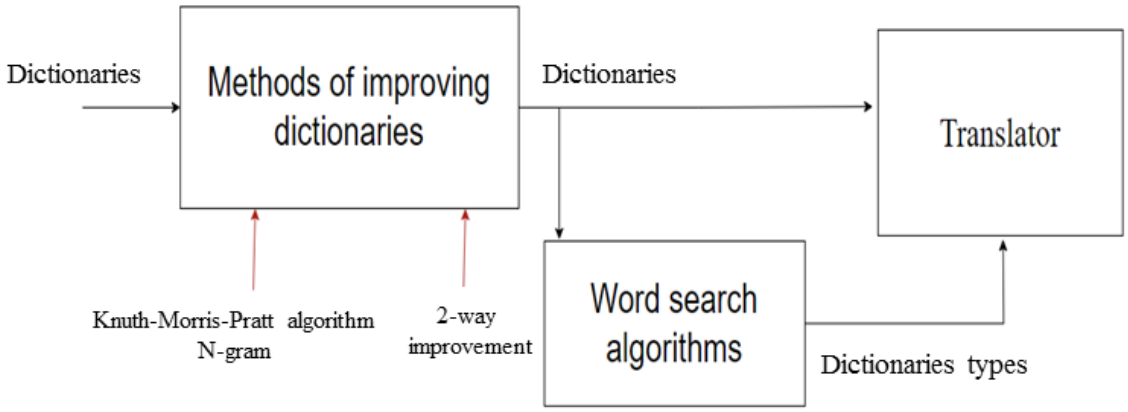


Fig.5. Interaction of components

For a translator to work quickly, you need to be able to quickly find a word in the dictionary. Consider such search methods as binary search, prefix tree, two pointers, balanced tree, root decomposition, and full traversal.

Since our dictionaries are sorted, we can apply the binary search method quite easily.

Binary search is an algorithm for finding a given value in an ordered array, which consists in comparing the middle element of the array with the desired value, and repeating the algorithm for one or the other half, depending on the result of the comparison.

This method will help us find the necessary information quite quickly, but will not provide additional information that can be used later.

After the experiments, the following results were obtained (Table 3). We can see that the time increases proportionally with a large amount of input data, and the algorithm is quite efficient.

Table 3

Results of the running time of the binary search algorithm

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Operating time (seconds)	0	1	5	11	41

For the two-pointer method, you need to use the following idea. For each pair of words, you need to make a query from the second dictionary. Since the dictionary is already sorted, for a quick search for words, you need to sort the queries and then use the following idea.

Since the words are sorted, if one word in the list is lexicographically larger than the second word in another list, then all the words following it are also larger than the selected word. Therefore, if we put pointers at the beginning of both lists, and move the one that is lexicographically smaller, we will not miss a single pair of words. Since each word occurs only once in the dictionary, and not always in the queries, if the words are equal, the pointer will move in the query list.

After the experiments, the following results were obtained (Table 4). From this experiment, one can see that the time increases proportionally with a large amount of input data, and the algorithm is quite efficient.

Table 4

Results of the running time of the algorithm of two pointers

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Operating time (seconds)	0	1	5	17	63

However, with a large number of words, the algorithm slightly loses to binary search. This is possibly related to the implementation of the algorithm, because with large data, even a small constant in the complexity can increase the real time of the algorithm many times, although the complexity of both is the same.

For the next experiment, we used the Prefix tree method. The use of the Prefix tree method was described in detail in [19] , so in this work we will only use the obtained results for comparison (Table 5).

Table 5

Results of the running time of the improved prefix tree

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Operating time (seconds)	0	1	3	10	33

Hashing algorithm is the next researched algorithm. For this algorithm, we will generate a hash function from each line, and distribute all the words by it, after which the obtained results will be searched by binary search. However, for our testing, let's eliminate the second part and just look at the speed of hashing for our strings.

Table 6

Results of the running time of the hashing algorithm

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Operating time (seconds)	0	0	0	2	6

From this experiment, we can see that hashing takes quite a bit of time, but it cannot be the only method of finding words, since two different strings can have the same hashes, and there will be false positives. This algorithm will allow you to mix the search area, but will increase the required amount of memory, which can be critical with a large amount of input data, so it should be used only when necessary (Table 6).

A balanced tree in the general sense of the word is a type of binary search tree that automatically maintains its height, that is, the number of levels of vertices under the root is minimal.

For this algorithm, we will use a modification of the usual map, where the keys will be our lines.

After the experiments, the following results were obtained (Table 7).

Table 7

Results of the running time of the balanced tree algorithm

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Operating time (seconds)	2	11	40	153	579

From this experiment, we can see that the time with a large amount of input data is very high compared to other algorithms. Since the algorithm has the same O(N*logN) complexity as the others, this is a rather unexpected result. Since we use a string as a key, their comparison takes quite a lot of time, which leads to such time consumption. However, can we say that the algorithm is not working at all? To do this, let's compare it with those that will have a worse difficulty, and compare it with them.

For the next algorithm, we used SQRT decomposition. Unlike the previous ones, it will have worse complexity. We will use the sqrt decomposition algorithm. To do this, in the sorted array, we will select the words that are in the positions $0, \sqrt{n}, 2\sqrt{n}, .., n$ containing approximately the root of n words, where n is the number of words in the list. Since the entire list has been sorted, the list selected in this way will also be sorted. Then, after comparing the input word with each of the words in the list, we can find the root of the length of the list of comparisons, on which the searched word can be found. Since we chose words with a different root between them, the length of the found subsegment will be equal to the root of n. Thus, for two roots of the length of the list of comparisons, we can find our word. After the experiments, the results were obtained (Table 8).

Table 8

Results of the running time of the sqrt decomposition algorithm

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Operating time (seconds)	45	225	1464	9516*	60000*

Since with the input data size of 10^7 words, the working time approaches 25 minutes, it was decided not to conduct an experiment for larger data, but to find the approximate working time of the algorithm. As you can see, the last experiment will take more than 16 hours of time, which is certainly not possible within the scope of this research.

It is clear that within the framework of this problem, the first thing you can do is simply to fulfill the condition, and for each of the words iterate over every other word – it is so called full search.. Since it will be quite long, instead of conducting an experiment, we will calculate the expected time of the algorithm. As a basis, we will take the operating time of the root decomposition algorithm, since it is the closest to it, and multiply it by the root of the size of the input data.

This is possible because the real time of the algorithm is equal to the number of commands that the algorithm will execute, divided by the frequency of the processor, or multiplied by the execution time of one command. Since we are calculating for the same processor, when one is divided by another, this value will decrease, and the ratio of the number of commands will be proportional to the complexity. Since $n^2 / n\sqrt{n} = \sqrt{n}$, we can multiply the root decomposition result by the root of the input data. After the calculations, the following results were obtained (Table 9).

Table 9

Results of the running time of the quadratic search algorithm

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Working hours (hours)	12.5	108	1286	14478	19 years old

From this experiment we can see that in order to process a large amount of data, we need to use at least some algorithms that will improve the usual choice of correspondence for each of the words
To compare the algorithms, we will summarize all the data obtained during the experiments into a common table (Table 10, Figures 6,7).

Table 10

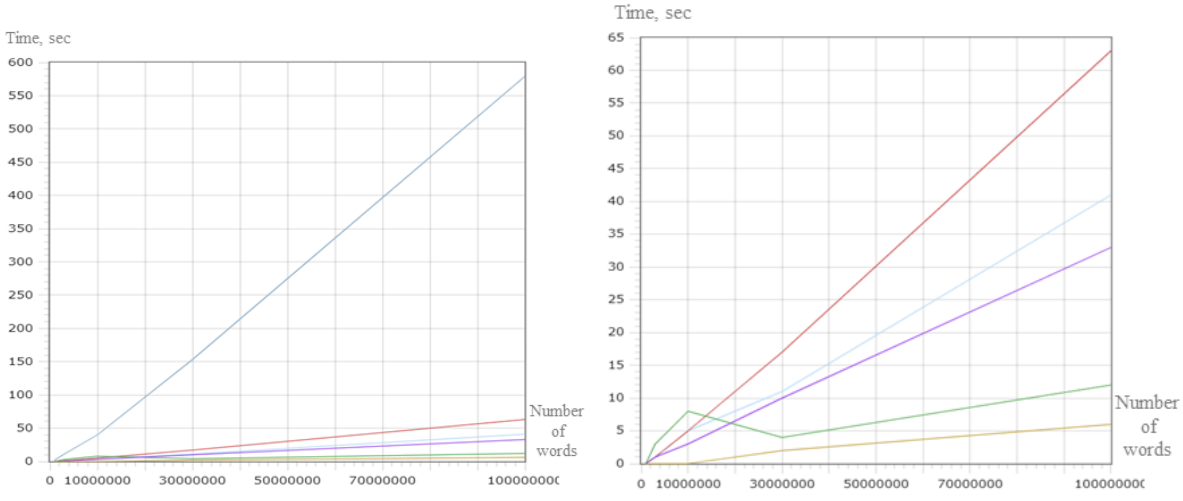
Comparison of algorithms

Size of input data (number of words)	10^6	3*10^6	10^7	3*10^7	10^8
Binary search	0	1	5	11	41
Two pointers	0	1	5	17	63
Prefix tree	1	3	8	4*	12*
Improved Prefix tree	0	1	3	10	33
Hashing	0	0	0	2	6
A balanced tree	2	11	40	153	579
Root decomposition	45	225	1464	9516**	60000**
Full search	12.5 hours **	108 hours**	1286h**	14478 hours **	19 years old **

*-The data was executed on a tree built on 10^6 words instead of the required ones.
**-Estimated expected working time.

After all experiments, it is clear that fast algorithms must be used for big data. The best choice would be either an improved boric, which works better with small alphabets, or a binary search, because they are fast and don't have the problem of false positives. The worst among the logarithmic algorithms is the balanced tree, so it is possible to use any other logarithmic algorithm depending on other factors, such as reliability, memory, and dictionary structure.

Graphs on Figure 6a and 6b show the runtime dependency on number of words in the dictionary. On Figure 6a one can see grey line, that corresponds to balanced tree algorithm. This graph was added to make the result obtained more demonstrative. But since it is difficult to analyze other results, Figure 6b shows the same dependencies, but without the balanced tree algorithm.



a) **b)**

Fig.6. Comparative graph of the runtime dependency on number of words in the dictionary for logarithmic algorithms: two pointers algorithm – red line; binary search algorithm- blue line; improved prefix tree algorithm – violet line; hashing algorithm- green line; prefix tree algorithm- brown line; balanced tree - grey line.

Conclusions

In the course of the work, an analysis of the machine text translation system for artificial languages and the augmentation methods used in it was carried out.

The method of storing dictionaries was improved using the following methods: storing words in the order in which they are presented; storing words in alphabetical order; balanced tree; hash table; root decomposition; prefix tree.

In addition, the methods of searching for elements in the dictionary were analyzed, such as quadratic search, root decomposition, prefix tree, two pointers, binary search and hashing. As a result, hashing, prefix tree and binary search showed the best result. Each of which has its own drawbacks, and we cannot clearly indicate the best method. The use of these methods made it possible to increase the speed of work on a large amount of data compared to a conventional search from several years to tens of seconds. This made it possible to use the translator on large volumes of text. When switching from the balanced tree algorithm to other logarithmic algorithms, the running time decreased by about 20 times.

The model has quite a high potential for use in cases where there are no conventional translators. With a growing number of languages, this problem may worsen, as the number of translators cannot grow at the same rate.

References

1. K. Manuel, KV Indukuri and PR Krishna, "Analyzing Internet Slang for Sentiment Mining," 2010 Second Vaagdevi International Conference on Information Technology for Real World Problems, 2010, pp. 9-11, doi: 10.1109/VCON.2010.9

2. F. Ren and K. Matsumoto, "Semi-Automatic Creation of Youth Slang Corpus and Its Application to Affective Computing," in IEEE Transactions on Affective Computing, vol. 7, no. 2, pp. 176-189, 1 April-June 2016, doi: 10.1109/TAFFC.2015.2457915

3. Karen S. Jones. Natural language processing: a historical review //Cambridge: Computer Laboratory, University of Cambridge, 2001

4.V. Ryzhkova, "Possibilities of Computer Lexicography in Compiling Highly Specialized Terminological Printed and Electronic Dictionaries (Field of Aviation Engineering)," 2020 Ivannikov Memorial Workshop (IVMEM), 2020, pp. 40-42, doi: 10.1109/IVMEM51402.2020.00013

5. B. Ranaivo-Malançon, S. Saee and JF Wilfred Busu, "Discovering linguistic knowledge by converting printed dictionaries of minority languages into machine readable dictionaries," 2014 International Conference on Asian Language Processing (IALP), 2014, pp. 140-143, doi: 10.1109/IALP.2014.6973522.

6. GR Chumarina. Classification of electronic dictionaries in modern lexicography and lexicologists and features of their use [Electronic resource] Baltic Humanitarian Journal. 2013. No. 4. P.123-126

7. Deng, Yonggang, and William Byrne. "HMM word and phrase alignment for statistical machine translation." IEEE Transactions on Audio, Speech, and Language Processing 16.3 (2008): 494-507.

8. .Hey, Xiaodong. "Using word-dependent transition models in HMM-based word alignment for statistical machine translation." Proceedings of the Second Workshop on Statistical Machine Translation . 2007.

9..Oh, Franz Josef. Statistical machine translation: From single word models to alignment templates . Diss. Aachen, Techn. Hochsch., Diss., 2002, 2003

10. Östling, Robert, and Jörg Tiedemann. "Efficient word alignment with markov chain monte carlo." The Prague Bulletin of Mathematical Linguistics (2016).

11. Moore, Robert C. "A discriminative framework for bilingual word alignment." Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. 2005.

12.Pitrelli, John F., et al. "The IBM expressive text-to-speech synthesis system for American English." IEEE Transactions on Audio, Speech, and Language Processing 14.4 (2006): 1099-1108.

13. Casacuberta, Francisco, and Enrique Vidal. "GIZA++: Training of statistical translation models." Retrieved October 29 (2007): 2019.

14. Olesia Barkovska , Oleg Mikhal , Daria Pyvovarova , Oleksii Liashenko , Vladyslav Diachenko , Maxim Volk . Local Concurrency in Text Block Search Tasks. // International Journal of Emerging Trends in Engineering Research. - Volume 8. No. 3, March 2020. – P.690-694. DOI: 10.30534/ijeter/2020/13832020

15.Croft, William. "Intonation units and grammatical structure." (1995): 839-882.

16. Kazakov D. Artificial naturalness // Science and life. – 2017. – No. 10. – P. 100–107.

17.D. Anggreani, DPI Putri, AN Handayani and H. Azis, "Knuth Morris Pratt Algorithm in Enrekang-Indonesian Language Translator," 2020 4th International Conference on Vocational Education and Training (ICOVET), 2020, pp. 144-148, doi: 10.1109 / ICOVET50258.2020.9230139.

18..Barkovska O., Pyvovarova D. and Serdechnyi V., Pryskorenyj alghorytm poshuku sliv-obraziv u teksti z adaptyvnoju dekompozycijeju vykhidnykh danykh. [Accelerated word-image search algorithm in text with adaptive decomposition of input data]. Systemy upravlinnja, navighaciji ta zv'jazku 4 (56), 28-34. (in Ukrainian) DOI: https://doi.org/10.26906/SUNZ.2019.4.028

19. Havrashenko A., Barkovska O. ANALYSIS OF TEXT AUGMENTATION ALGORITHMS IN ARTIFICIAL LANGUAGE MACHINE TRANSLATION SYSTEMS //Advanced Information Systems. – 2023. – No. 7. - no. 1. – pp . 47-53. DOI: 10.20998/2522-9052.2023.1.08

20. Olesia Barkovska, Anton Havrashenko, Vladyslav Kholiev, Olena Sevostianova. Automatic text translation system for artificial languages. // Computer systems and information technologies. – 2021. – no. 3. – p . 21-30. DOI: 10.31891/CSIT-2021-5-3

Oleksandr Melnychenko **Олександр Мельниченко**	PhD student of the Department of Computer Engineering and Information Systems, Khmelnytskyi National University, Khmelnytskyi, Ukraine. e-mail: oleksandr.melnychenko@live.com https://orcid.org/0000-0001-8565-7092	аспірант кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет, Хмельницький, Україна.