Computing and Informatics, Vol. 37, 2018, 1443–1463, doi: 10.4149/cai_2018_6_1443

METHOD TO OVERCOME THE AMBIGUITIES IN SHALLOW PARSE AND TRANSFER MACHINE TRANSLATION

Jernej Vičič

University of Primorska, The Andrej Marušič Institute
Muzejski trg 2, 6000 Koper, Slovenia
&
Research Centre of the Slovenian Academy of Sciences and Arts
The Fran Ramovš Institute
e-mail: jernej.vicic@upr.si

Marko Grgurovič

University of Primorska, The Andrej Marušič Institute
Muzejski trg 2, 6000 Koper, Slovenia
e-mail: marko.grgurovic@student.famnit.upr.si

Abstract. The manuscript proposes a new architecture for a Shallow Parsing and Shallow Transfer Rule-Based Machine Translation System. The newly proposed architecture omits the disambiguation module in the starting phases of the translation pipeline and stores all translation candidates generated by the ambiguous process in the morphological analysis phase. The architecture is based on multigraphs. We propose a simplified version of k-best shortest path algorithm for this special case of directed multigraph. The k-best set is ranked using a trigram language model. The empirical evaluation shows that the new architecture produces better translation quality results with constant delay in time.

Keywords: Multigraph, k-shortest paths, shallow parse and transfer RBMT, machine translation

Mathematics Subject Classification 2010: 68T50

1 INTRODUCTION

The manuscript proposes a change in the architecture for a Shallow Parsing and Shallow Transfer Rule-Based Machine Translation System. One of the methods, which guarantees relatively good results for the translation of closely related languages, is the method of a rule-based shallow-parse and shallow-transfer approach which uses a simple architecture, thus relying on (mostly structural) similarity of the language pair. It has a long tradition and it has been successfully used in a number of MT systems, the most notable of which are Apertium [3] and Česílko [11].

Shallow-transfer systems usually use a relatively linear and straightforward architecture where the analysis of a source language is usually limited to the morphemic level. Most such systems still rely on morphological analysis of the source text in the source language. This part of the process is ambiguous. The newly proposed architecture omits the disambiguation module in the starting phases of the translation pipeline and stores all translation candidates generated by the ambiguous process in the morphological analysis phase.

The time and space complexity of the proposed architecture are discussed along with the presentation of the algorithms and data structures. An experimental prototype system as a proof of concept has been constructed on the basis of the Apertium [3] machine translation framework and using language data for the language pair Slovenian-Serbian [14].

Neural machine translation (NMT) has recently replaced the "classical statistical machine translation – SMT" and becomes the dominant research paradigm [1], but there are still reports of RBMT systems outperforming SMT or NMT systems such as [14] and also there are use cases where RBMT systems are more suitable (where the deterministic behaviour of the system is important).

The rest of the manuscript is organised as follows: The domain description is presented in Section 2, the motivation for the research is presented in Section 3. The methodology is presented in Section 4, space and time complexity of the presented data structures and algorithms is presented in Section 5, empirical evaluation and results are presented in Section 6 and conclusions in Section 7.

2 DOMAIN DESCRIPTION

The European Association for Machine Translation (EAMT) [5] describes Machine Translation as any translation task that involves the use of computers (machines). In the scope of this paper we will use the term in a narrower scope: Fully Automatic Machine translation [5] where the task of translating the text from the source language to the target language is done by the computer (machine). More specifically the research focuses on the translation of similar languages. One of the most suitable paradigms for this domain is the Shallow Transfer Rule-Based MT (ST-RBMT).

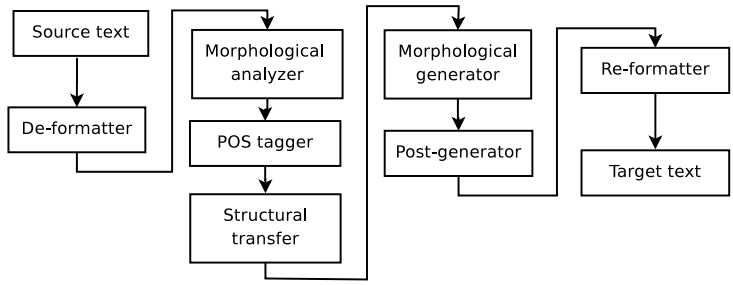

Figure 1. The modules of a typical shallow parsing and shallow transfer translation system. The frameworks for the construction of MT systems [3, 12, 17, 21] follow this design.

2.1 Shallow Parsing and Shallow Transfer Rule-Based MT

Figure 1 shows the architecture of the most known translation systems for related languages Apertium [3] and Česílko [12]. The newest version of Česílko [26] is available under open source license.

The monolingual dictionaries are used in the morphological parsing of the source text by the morphological analyser module and in the generation of the translation text in the target language by the morphological generator module. The Part Of Speech (POS) tagger module is used to disambiguate the ambiguous output of the morphological analyser module. The bilingual dictionary is used for word-by-word translation: in our case the translation is based on lemmata. The shallow transfer rules are used to address local syntactic and morphological rules such as local word agreement and local word reordering. The module using the bilingual dictionary and the shallow transfer rules is the structural transfer module. The remaining modules deal with text formatting which is not the domain of this paper. All methods and materials discussed in this paper were tested on a fully functional machine translation system based on GUAT [24], a translation system for related languages based on Apertium [3], which is a widely used open source toolkit for creating machine translation systems between related languages.

The majority of the translation systems for related languages use the shallow parsing machine translation architecture [25].

2.1.1 Apertium

Apertium is an open-source machine translation platform, initially aimed at related-language pairs but recently expanded to deal with more divergent language pairs (such as English-Catalan). The platform provides a language-independent machine translation engine, tools to manage the linguistic data necessary to build a machine translation system for a given language pair and linguistic data for a growing number of language pairs. All these properties make Apertium a perfect choice in a cost-effective machine translation system development.

2.1.2 GUAT

All methods and materials discussed in this paper were tested on a fully functional machine translation system based on GUAT [14] and [24], a translation system for related languages based on Apertium [3]. The system GUAT was used as the sandbox for the implementation of proposed methods. GUAT is automatically constructed so there is still a room for improvement, mainly through data correction tasks. The basic architecture of the system follows the architecture of Apertium [3] and is presented in Figure 1.

2.2 Multigraph

In mathematics, a graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related". The objects correspond to mathematical abstractions called vertices (also called nodes or points) and each of the related pairs of vertices is called an edge (also called an arc or line). A graph G is an ordered pair $G := (V, E)$ with:

- V a set of vertices or nodes,
- E a set of unordered pairs of vertices, called edges or lines.

A multigraph or pseudograph is a graph which is permitted to have multiple parallel edges between nodes, that is, edges that have the same start and end nodes. Thus two vertices may be connected by more than one edge. Formally, a multigraph G is an ordered pair $G := (V, E)$ with:

- V a set of vertices or nodes,
- E a multiset of unordered pairs of vertices, called edges or lines.

In our example we use a subset of the above definition, a directed multigraph, where the edges defined by the pairs of the E multiset are ordered (directed from start to finish). Another addition to the typical definition of a graph for our example was the addition or a starting node.

The new definition of a directed multigraph with a starting node used in the paper is:

- V a set of vertices or nodes,
- E a multiset of ordered pairs of vertices, called edges or lines,
- s a starting node.

Such multigraphs allow compact description of all available translation hypotheses.

2.3 k-Shortest Paths

A problem related to the well-known single-source shortest path problem is the k-shortest paths problem. In the latter we are tasked with finding not only the shortest path to a given vertex, but up to k-shortest paths. There are many variations on this theme: depending on the structure of the graph; whether we are interested in only a specific s–t path or multiple such pairs; or even if the paths ought to be loopless or not. There exist efficient algorithms for computing these paths [6]. A variant of the k-shortest paths algorithm, which we will describe later, was implemented and used to construct a k-best set of translation candidates.

3 MOTIVATION

The shallow-transfer RBMT architecture usually relies on disambiguation after the morphological analysis to cope with the possible multiple outcomes. Figure 1 shows the disambiguation module following the morphological analysis. This process can be done by a set of rules in a form of a Constraint Grammar [15] using Visl[1] as used in [20] or in most cases by using a statistical POS tagger such as [18]. This phase precedes the more or less deterministic transfer phase. This is obviously a huge limitation, especially for the lexical transfer, since in most language pairs there are many words where translation depends upon the syntactic and/or semantic context. If the system contains some (shallow) syntactic parser and/or structural transfer, they also tend to produce ambiguous output relatively often.

The most important reasons for this research are:

- The production of a new POS tagger, especially a good quality tagger, is not a simple task. One of the easiest methods is the training of a stochastic tagger based on HMM algorithm [27]. Some parts of this task can be automatized using unsupervised learning methods or supervised learning methods like [2], but it still involves the selection of a new tag set, the production of a tagged training corpus, testing of the corpus and, at the end, the basic learning process.
- The quality level of the tagging process of today's state-of-the-art POS taggers for highly inflectional languages like Czech and Slovak [10] and Slovenian, Croatian and Serbian [8] is relatively low, comparing to the quality of POS taggers for the analytical languages like the English language, and also comparing to the overall quality of the translation systems for related languages.
- According to the today's most used designs for translation systems for related languages, the shallow transfer translation systems, the disambiguation module follows the source language morphological analysis at the beginning of the translation process. This design is shown in Figure 1. Such a design is adopted by Apertium [3] and Česílko [12]. Errors produced at the early stages of the transla-

[1] `http://visl.sdu.dk/constraint_grammar.html`

tion process usually cause bigger problems than errors introduced at later phases as later phases of the translation rely on the output of the preceding phases.

- Multiple translation candidates allow selection of the best candidates in the final phase when all available data for the translation has been accumulated. The most common translation errors are fluency errors of the target language and not adequacy errors. These errors usually do not interfere with the meaning of the translation but rather with the grammatical correctness of the translation. They are mostly caused by the errors in morphological analysis or morphological syntheses.

The omission of the tagger and introduction of a ranking scheme based on target language statistical model, as suggested in [13], yields better translation results. The introduction of multiple translation candidates generated from all possible morphological ambiguities, as suggested in [13], leads to an exponential growth of the number of possible translation candidates.

The paper [25] proposed a rule-based method for eliminating the impossible translation candidates, thus lowering the number of possible translation candidates. A statistical ranking method was used to select an arbitrary number of best candidates. The rules were automatically constructed.

The method proposed in this paper keeps all possible translation candidates in the starting phases of the translation process, all the candidates are considered and the best candidate (or n-best set of candidates) is selected in the last phase, the ranking phase. The ranking phase uses a standard statistical language model to score possible translation candidates.

4 METHODOLOGY

4.1 Proposed Architecture

The unified data structure would result in the rewrite of almost all modules of the original Apertium system. One of the most appealing features of the Apertium system is the transparency of the translation process. All data is shared in a human-readable text form through simple UNIX pipes resulting in easy error discovery and easy debugging. The proposed data structure would have to be serialized in a human-readable form.

No change was made to the Apertium toolset for the means of the presented experiment. A new module has been added to the architecture, the Multigraph supervisor, which constructs the multigraph data structure and communicates with the Apertium modules through UNIX named pipes. The new architecture is presented in Figure 2. The Multigraph supervisor module constructs the translation candidates by sending parts of the sentences, connecting edges in the data structure, to the appropriate module and saves the result in the same data structure gradually constructing all translation candidates. The data structure is further presented in Section 4.1.1.

The POS tagger module that handles the morphological disambiguation has been omitted from the architecture as all the ambiguities are stored and dealt with in the later modules.

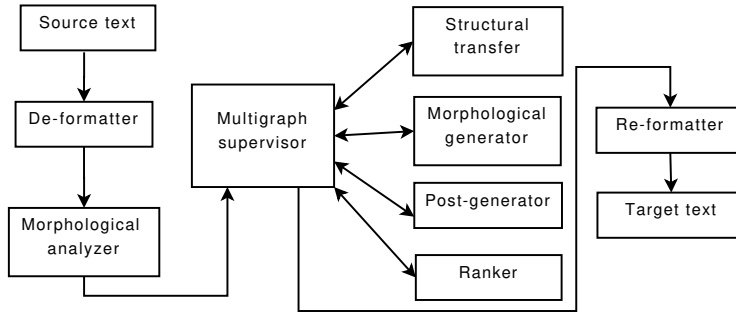

Figure 2. The proposed new architecture of a shallow transfer RBMT system using multigraphs. The Supervisor module uses the original modules in the translation process.

4.1.1 The Data Structures

Two data structures based on multigraphs were used in the Multigraph supervisor module: the construction process and the most distinct properties are presented in the subsections following this section.

The pilot implementation of the presented architecture was done in Java using the Java Universal Network/Graph Framework – JUNG, which is an open-source software library that provides a common and extendible graph/network analysis and visualization framework. JUNG also provides a visualisation framework that makes it easy to construct tools for the interactive exploration of network data.

4.1.2 Morphological Analysis

The morphological analysis produces ambiguous results. There are multiple Morpho-Syntactical descriptors – MSD [7] that can be attributed to one word form; an example of an ambiguously tagged sentence is presented in Figure 3. The MSD tags used in this example are:

- ⟨adv⟩ – adverb,
- ⟨vbser⟩ – auxiliary verb to be,
- ⟨pres⟩ – present tense,
- ⟨p3⟩ – third person,
- ⟨sg⟩ – singular,
- ⟨vblex⟩ – regular verb,
- ⟨f⟩ – female gender,

- ⟨acc⟩ – accusative case,
- ⟨nom⟩ – nominative case,
- ⟨pos⟩ – positive,
- ⟨nt⟩textgreater – neuter gender,
- ⟨n⟩ – noun.

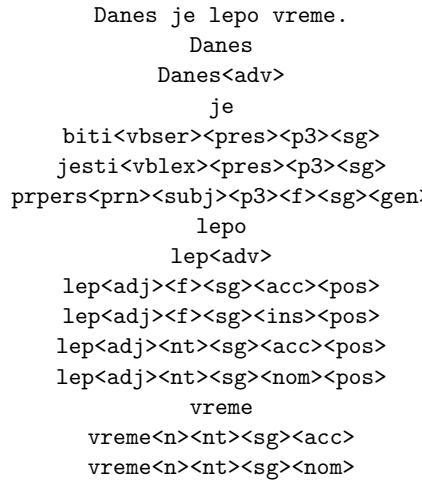

Figure 3. The ambiguously tagged sentence *Danes je lepo vreme*

The introduction of multiple translation candidates generated from all possible morphological ambiguities, as suggested in [13], leads to an exponential growth of the number of possible translation candidates.

The output of the morphological analysis is a set of all possible morphological tags describing each word. Every word with more than one tag can be observed as a set of possible ambiguities. In the case of highly inflectional languages like the pair presented in this paper the number of ambiguous possibilities increases. The set of all possible translation candidates is constructed as the vector product of all ambiguous sets. The number of possible translation candidates grows exponentially with the length of the sentence, the upper limit of the number of possible translation candidates is: $\prod_{i=0}^{|S_{max}|} x_{i_{max}}$, where S_{max} is the longest sentence and $x_{i_{max}}$ is the biggest number of ambiguities for a word. The average number of possible translation candidates is much lower, it is $\bar{x}^{\bar{S}}$, where $\bar{x}$ is the average number of ambiguities and $\bar{S}$ is the average length of a sentence.

The following example shows empirical values for an example source sentence and typical numbers collected from a corpus test-set: the maximal values: $|S| = 40$, $x = 15$, $|TC| = \prod_{i=0}^{40} 15 = 110.573323209e{+}45$ and the average values: $\bar{S}| = 15$, $\bar{x} = 3$, $|TC| = \prod_{i=0}^{15} 3 = 14\,348\,907$.

The data structure that can contain all the information produced by the morphological analysis in a compact form and also enable easy access to all translation candidates is a multigraph, where nodes represent word boundaries and edges represent all possible ambiguous word forms. An example multigraph of the same example sentence from Figure 3 is shown in Figure 4.

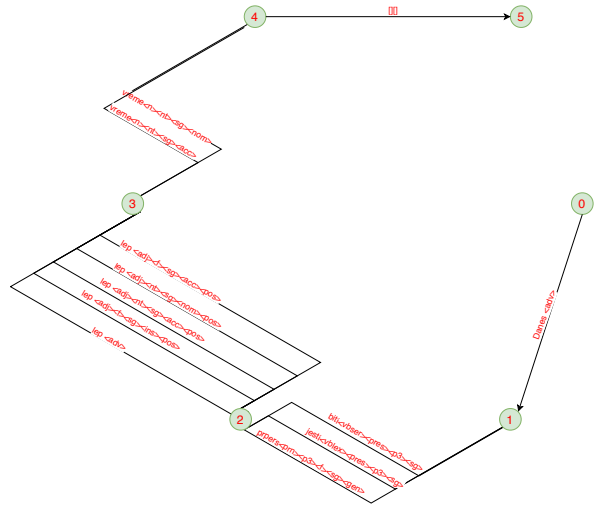

Figure 4. The multiple possibilities of the morphological analysis are stored in the edges of the multigraph. The example multigraph is constructed from the data in Figure 3.

4.1.3 Structural Transfer

The structural transfer rules are usually made in two parts: the search pattern (context) and action. We will concentrate on Apertium style rules although the abstraction would apply to most systems. A search through the Apertium systems[2] showed that the length of 3 elements for the search pattern suffices for more than 98 % of rules. The linguistic explanation is that the rules act in a very limited context.

Although we can safely use the length of the longest context (search pattern) of 3 in the majority of cases, we will abstract the length to an arbitrary length l_R. All possible candidates of the length l_R are constructed starting at the beginning of the multigraph – an example is shown in Figure 4 – and gradually moving to the last node of the multigraph. This technique enables the applications of the rules in the Left to Right Longest Match (LRLM) order which has been proven to be effective by [23].

[2] Apertium project at Sourceforge: `http://sourceforge.net/projects/apertium/`

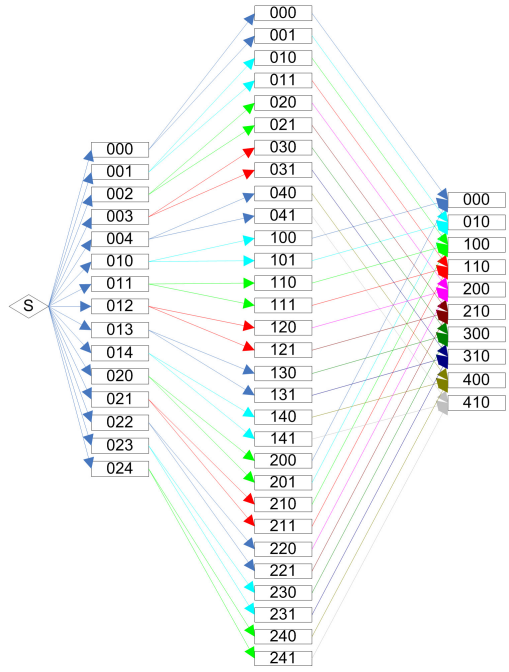

Figure 5. The data structure storing all the data from the Structural transfer module

It can be easily proven that this algorithm constructs an LRLM coverage of all translation candidates. The candidates are sent to the structural transfer module and the result is stored in a new data structure shown in Figure 5. The value for l_R has been set to 3 to simplify the visualization of the data structure.

Figure 5 shows a representation of the complex data structure produced from the morphological output, presented in Figure 3, and stored in the multigraph presented in Figure 4. All the morphological descriptors have been numbered. The paths connecting the morphological descriptors have been constructed using these numbers. Let us observe the examples in Figure 6 where the first two trigrams of the lexical units are represented by the strings *"000"* and *"001"*.

```
Danes je lepo vreme.
Danes<adv> biti<vbser><pres><p3><sg> lep<adv>
000
Danes<adv> biti<vbser><pres><p3><sg> lep<adj><f><sg><acc><pos>
001
```

Figure 6. The first two trigrams represented by the strings *"000"* and *"001"*, respectively

All strings of a certain length are stored in the same column. The trigrams containing the morphological descriptors of three adjacent word forms are stored in nodes: the edges will be used to store the probabilities of the trigrams.

4.1.4 Morphological Generation

The lexical units that were stored as n-grams in the complex data structure are fed to the Morphological generator which generates the linearized text. The data is stored in the same data structure.

4.1.5 Ranking

The ranking process simulates the Statistical N-gram Language Models, in particular the Trigram Language Model. The Trigram Language Model is the most used statistical language model. Probability of a string $P(S)$ is represented by Equation (1), where w_i denotes the i^{th} word of the string S. The probability of a string S is equal to the product of the conditional probabilities of the words constituting the set, with the condition that all the previous words of the string S appear before the i^{th} word.

$$\begin{aligned} P(s) &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\dots p(w_l|w1\dots w_{l-1}) = \\ &= \prod_{i}^{l} P(w_i|w_1\dots w_{i-1}). \end{aligned} \tag{1}$$

The process computes the probabilities for all trigrams and stores them in the multigraph data-structure. The probability of the observed trigram is stored in the edge finishing in the observed node.

The problem of ranking the best translation candidate using the trigram language model based on the presented data structures becomes the search for the minimal path in the graph from a starting node to the finishing node. The most known algorithm that can be used is the Dijkstra algorithm [4]. An algorithm that produces k-shortest paths must be used in order to produce an n-best-set, actually k-best-set according to the presented nomenclature, of translation candidates. The graph presented in Figure 5, that presents the final data structure with candidates for the final translation, is an acyclic, directed (multi)graph.

Since the essentially optimal algorithm given in [6] is somewhat involved, we present a simplified method for computing up to k-shortest paths in directed acyclic graphs (DAGs) and provide the straightforward analysis. In DAGs, the question of whether the computed paths are loopless or not is moot: there are no loops. We will denote the out-degree of a vertex $v \in V$ by $\deg^-(v)$ and the cost of an edge $(u, v) \in E$ by $\ell(u, v)$. Furthermore, we use the following convention: $n = |V|$ and $m = |E|$.

The algorithm works by traversing the set of vertices in reverse topological order, which can be obtained in linear time, starting with the target vertex t. Each vertex

keeps a list of up to k distances of shortest paths to t, which is initially empty, except for t which contains a distance of 0. Then, each vertex i loops through its outgoing edges (i, j) and inserts into a priority queue each endpoint j, with a priority equal to the first element in the list of j plus the cost of the edge (i, j). Once this is done, each vertex pops up to k elements from the priority queue and inserts them into its list. After an element is popped, a new one is inserted into the priority queue from the same list as the popped element. Formalizing in pseudocode we get Algorithm 1.

Algorithm 1 Proposed k-shortest path algorithm for DAGs

```
procedure k-SHORTEST-PATHS(V, E, k, t)
    paths := array of n lists
    paths[t] := empty list
    paths[t].append(0)
    for all v ∈ V \ {t} do                    ▷ In reverse topological sorted order
        paths[v] := empty list
        PQ := empty
        pointers := array of deg⁻(v) list pointers
        for all (v, j) ∈ E do
            pointers[j] := paths[j]
            PQ.enqueue(j, pointers[j].data + ℓ(v, j))
        end for
        count := 0
        while count < k ∧ ¬PQ.empty do
            j := PQ.deleteMin()
            paths[v].append(pointers[j].data + ℓ(v, j))
            count := count + 1
            if pointers[j].next ≠ null then
                pointers[j] = pointers[j].next
                PQ.enqueue(j, pointers[j].data + ℓ(v, j))
            end if
        end while
    end for
end procedure
```

The time and space complexity of the presented algorithm is presented in Section 5.1.

5 SPACE AND TIME COMPLEXITY

A few definitions that will alleviate the discussion of the complexity of the presented algorithms and data structures.

- l_R – longest rule pattern length,

- l_S – longest sentence length,
- a – biggest number of ambiguities.

In numbers: setting the longest sentence to 30 words and the biggest number of ambiguities for a word to 36 (from the corpus Multext-east [9] and the GUAT system morphology [22]). The average number of ambiguities is 3. Setting the longest rule to 3 as more than 98 % of all the rules in Apertium systems and all the rules in the GUAT system have 3 or fewer lexical units in the pattern.

The data structure presented in Section 4.1.2 is a multigraph with l_S nodes and a edges between two nodes. The number of edges m is defined as: $m = l_S \cdot a$ and giving the maximum and minimum number of edges: $m_{max} = 30 \cdot 36 = 1\,080$ and $m_{average} = 30 \cdot 3 = 90$, respectively. The average number of edges in our example setting is $m = l_S \cdot a = 30 \cdot 14 = 780$ and the number of nodes is $n = 30$.

The data structure presented in Section 4.1.3 is a multigraph with $l_S - l_R + 1$ sets of nodes, where each set of nodes can have up to a^{l_R} nodes. Each node is connected with up to a edges (valence) to nodes in the adjacent set of nodes.

Equations (2) and (3) present the total number of nodes and edges and the continuation presents the empirical projections of the presented values using maximal and average values from the corpus Multext-east [9], and Equation (3) presents the total number of edges using the same values, as presented in the previous example.

$$\begin{aligned} n &= a^{l_R} \cdot (l_S - l_R + 1), \\ n_{max} &= 36^3 \cdot 28 = 1\,306\,368, \\ n_{average} &= 3^3 \cdot 28 = 252; \end{aligned} \tag{2}$$

$$\begin{aligned} m &= na, \\ m_{max} &= a^4 \cdot 28 = 36^4 \cdot 28 = 47\,029\,248, \\ m_{average} &= a^4 \cdot 28 = 3^4 \cdot 28 = 2\,268. \end{aligned} \tag{3}$$

The worst-case scenario cannot be reached as most of the word positions and POS variations are dependent.

The construction of the graph, basically the morphological analysis, and the execution of the structural transfer process are linear to the number of edges, so the time complexity can be attributed mostly to the largest contributor, the ranking process.

5.1 The Ranking Process

Analysis is as follows. In the first stage, each vertex $i \in V$ goes through its outgoing edges and places the first element found in each of its neighbors' lists into the priority queue. It is not difficult to implement the priority queue insert operation in time $O(1)$, and using adjacency lists the loop through the neighbors can be done in $O(\deg^-(i))$. We can write the cost for the first stage over all vertices as $O\left(\sum_{i=0}^{n} \deg^-(i)\right) = O(m)$ by the handshaking lemma.

In the second stage, each vertex $i \in V$ performs k deleteMin operations, and for each deleteMin operation it also inserts the next element from the list into the priority queue. The deleteMin operation can be performed in $O(\lg \deg^-(i))$, since there are at most $\deg^-(i)$ elements inside the priority queue at any given time. Finding the next element in the list and inserting it into the priority queue can be done in $O(1)$. Writing the cost over all vertices for this stage, we get the values presented in Equation (4):

$$\begin{aligned} O\left(\sum_{i=0}^{n} k \lg \deg^-(i)\right) &= O\left(k \lg \left(\prod_{i=0}^{n} \deg^-(i)\right)\right) \\ &= O\left(k \lg \left(\left(\frac{m}{n}\right)^n\right)\right) \\ &= O\left(nk(\lg m - \lg n)\right). \end{aligned} \tag{4}$$

Together with the first phase, this amounts to $O(m+nk(\lg m-\lg n))$. In general we can bound the number of edges by $m = O(n^2)$ and obtain $O(m + nk \lg n)$. We point out, that our time bound is never better than that of [6], which for DAGs amounts to $O(m+n+k)$. The space bounds of Algorithm 1 are simply $O(m+nk)$, since each vertex keeps a list of length at most k and the $\deg^-(v)$ pointers are simply $O(m)$ over the entire algorithm, by the handshaking lemma.

For our particular problem, we can bound the running time as follows, using the worst-case values we have computed in the previous section. Equation (5) presents the final values.

$$O\left(m + k \lg \left(\left(\frac{m}{n}\right)^n\right)\right) = O\left(m + k \lg \left(\left(\frac{na}{n}\right)^n\right)\right) = O\left(m + nk \lg(a)\right) \tag{5}$$

where a, the number of ambiguities, is a very small number, e.g. 3, as argued in the previous section. Therefore the log term is of no practical significance.

6 EMPIRICAL EVALUATION AND RESULTS

Two main goals were evaluated in this experiment:

- the change in the quality of the final results, the translations,
- the time complexity.

The newly proposed system was compared to two already available translation systems for the same language pair:

- the original off-the-shelf Apertium system with the Slovenian-Serbian translation data, described in [24],
- to the system presented in the experiment [25].

The later system uses a method to restrict the number of translation candidates.

6.1 Translation Quality Comparison

The Human-targeted Translation Edit Distance – HTER [19], which is derived from the edit-distance [16], was used to evaluate the translation quality. The metric counts the number of deletions, insertions and substitutions that need to be performed among the observed sequences, i.e., count the minimal number of edits needed to produce a correct target sentence from an automatically translated sentence. The definition of a correct translation understood in this experiment is a translation that is syntactically correct and expresses the same meaning as the source sentence and is achieved with the least amount of transformation. This procedure shows how much work has to be done to produce a good translation. The metric roughly reflects the complexity of the post-editing task.

Two evaluations were performed:

- The comparison between the system presented in this paper (GUAT Multigraph) and systems from a similar experiment [25] – the evaluation was performed on a small test-set (57 sentences);
- The translation quality evaluation using the same methodology as the former comparison using a bigger test-set (500 sentences).

The first evaluation was done on a relatively small test-set due to the constraints of the systems evaluated in the experiment [25]; one of the systems (GUAT all candidates) used all possible morphological ambiguities for the generation of translation candidates resulting in possibly millions of translation candidates. Such translations lasted hours and even days. The test data for this part of the experiment comprised of the 57 sentences. The sentences were chosen by length (sentences shorter than 15 words). This limitation still enabled a fair comparison of the translation quality of all the systems. The complexity of each sentence was arbitrary, there was no special selection of the sentences using this criteria although shorter sentences are usually simpler in structure. The results of this part of the evaluation are presented in Figure 7.

GUAT original – the reference system, based on Apertium architecture.

GUAT all candidates – a system that kept all translation candidates to the last phase (best translation performance, exponential growth of possible translation candidates).

GUAT rules selection – the system with a method that restricted the number of possible translation candidates in the starting phases of the translation.

GUAT Multigraph – the system with the newly proposed architecture.

The newly proposed system (GUAT Multigraph) mean value of HTER is 0.1712 which is better than the off-the-shelf system (GUAT original) mean value of HTER (0.2213). The paired two sample ttest for means in Table 1 shows that mean values are significantly different. The P value of the ttest is less than alpha (0.05), so

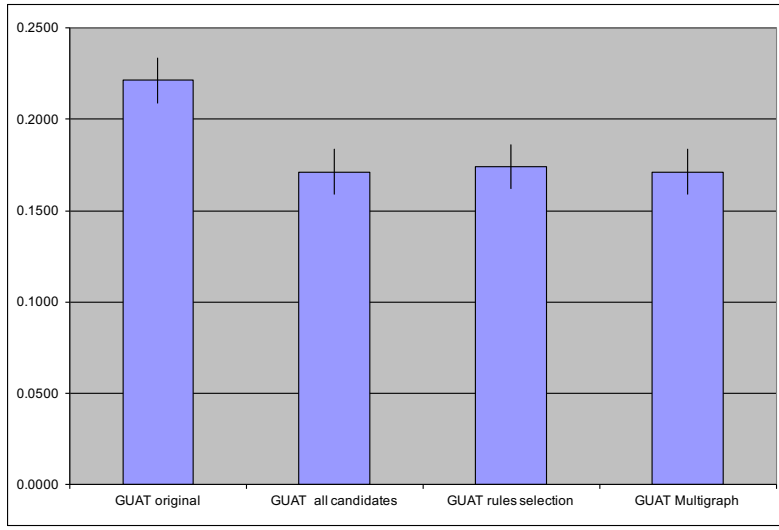

Figure 7. The translation quality evaluation, using HTER metric (less is better). The newly proposed system (GUAT Multigraph) outperforms the original system from [25] and equals the improved systems from the same experiment.

we can reject the null hypothesis (that there is no difference in mean) and we can accept that the mean value of GUAT Multigraph is lower than the mean value of the original system.

System	GUAT original	GUAT Multigraph
Mean	0.221269	0.171212
Variance	0.018006	0.013760
Observations	57	57
Hypothesis	no difference	
$P(T \leq t)$ two-tail	0.00011	

Table 1. The ttest shows that the mean values are significantly different

Table 2 shows that there is no significant difference in the mean values of GUAT rules selection and GUAT Multigraph. The values of the GUAT all candidates and GUAT Multigraph are the same, all translations were equal (both systems prepared and selected the same candidate translations).

The second evaluation was made to support the results of the first evaluation on a bigger test-set comparing the system presented in this paper (GUAT Multigraph) and the off-the-shelf GUAT system, which is based on the original Apertium architecture. The basic methodology was the same as in the first comparison. A new test-set was prepared which was comprised of 500 sentences randomly selected from the MULTEXT-East corpus [9]. Both systems used the same translation data, the

System	GUAT rules sel.	GUAT Multigraph
Mean	0.174157	0.171212
Variance	0.014498	0.013760
Observations	57	57
Hypothesis	no difference	
Alpha	0.05	
$P(T \leq t)$ two-tail	0.160599	

Table 2. The ttest shows that the difference in mean values could be coincidental (we cannot trust the difference)

construction process of the translation data was described in [24] and the data is available at the Sourceforge[3]. The results of this part of the comparison are presented in Figure 8. The system with the newly proposed architecture shows an improvement over the reference system.

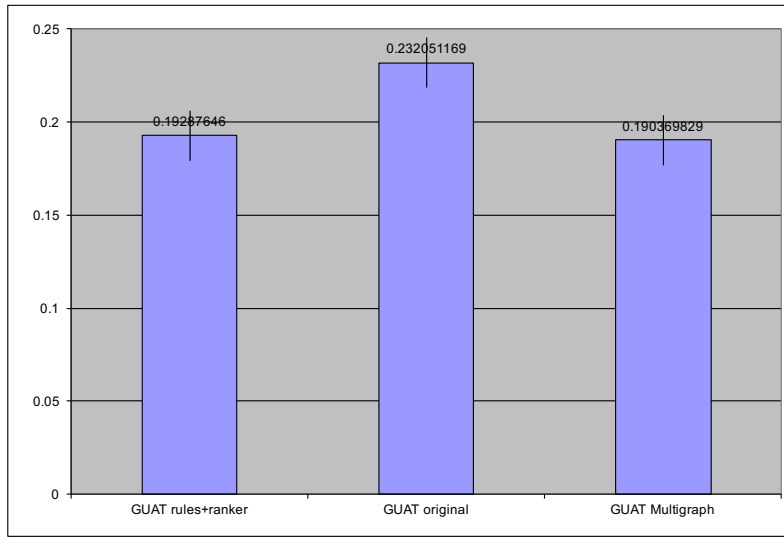

Figure 8. The translation quality evaluation, using HTER metric, of the original GUAT system based on the original Apertium architecture comparing to the newly proposed system. The newly proposed system outperforms the original system.

The ttest presented in Table 3 shows that the mean values are significantly different.

[3] `http://sourceforge.net/projects/apertium/`

System	GUAT original	GUAT Multigraph
Mean	0.232557	0.190251
Variance	0.020446	0.017484
Observations	200	200
Alpha	0.05	
Hypothesis	no difference	
$P(T \leq t)$ two-tail	0.00000000794	

Table 3. The ttest shows that the mean values are significantly different

6.2 Time Complexity

The empirical evaluation of the time complexity was done simultaneously with the evaluation of the translation quality. The test-data is described in Section 6.1, which was comprised of 200 sentences randomly selected from [9] corpus. The tests were performed on a personal computer[4]. Table 4 shows the time complexity comparison between the original GUAT system and the newly proposed system.

System:	GUAT	Multigraph
Nr. of sentences	200	200
Total time (seconds)	377.78	6 453.56
Per translation	1.89	22.27
Ratio	1.00	11.78

Table 4. Empirical evaluation of the time complexity. The newly proposed system is roughly 6 times slower than the original on the selected test sentences.

Description of Table 4:

GUAT – The reference system, based on [24].

Multigraph – the system with the newly proposed architecture.

Total time (seconds) – the total time the system spent to translate all 200 test sentences.

Per translation – the average time spent per translation.

Ratio – The ratio between the time spent by the reference system – GUAT and the described system.

7 CONCLUSIONS

The presented architecture represents a viable solution to the problem of exponential growth of the number of possible translation candidates in a non-disambiguated shallow transfer translation system. The empirical evaluation showed an improvement in the translation quality compared to the original system and also compared

[4] Laptop computer with Intel i3 CPU M330@2.13 GHz and 4 GB of memory

to the values presented in [25], which was an attempt to limit the number of possible translation candidates. The empirical evaluation also showed that the new system performed as well as a system that included all possible translation candidates, which shows that it always selected the best translation candidate.

The empirical evaluation of the time consumption showed that the new system performed roughly 12 times slower than the reference system and the constant factor was present in all test examples showing that the time complexity differed to a constant factor.

The proof-of-the-concept system has been implemented and it proved to be working as expected. A true implementation of the newly proposed architecture with the new module is already in progress. It could be incorporated into the Apertium framework.

REFERENCES

[1] Bojar, O.—Helcl, J.—Kocmi, T.—Libovick, J.—Musil, T.: Results of the WMT17 Neural MT Training Task. Proceedings of the Conference on Machine Translation (WMT), Volume 2: Shared Task Papers, 2017, pp. 525–533, doi: 10.18653/v1/w17-4757.

[2] Brants, T.: TnT – A Statistical Part-of-Speech Tagger. Proceedings of the 6th Applied NLP Conference, Seattle, WA, 2000, pp. 224–231, doi: 10.3115/974147.974178.

[3] Corbí-Bellot, A. M.—Forcada, M. L.—Ortiz-Rojas, S. et al.: An Open-Source Shallow-Transfer Machine Translation Engine for the Romance Languages of Spain. Proceedings of the EAMT Conference, 2005, pp. 79–86.

[4] Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, Vol. 1, 1959, No. 1, pp. 269–271, doi: 10.1007/bf01386390.

[5] EAMT. European Association for Machine Translation, 2018.

[6] Eppstein, D.: Finding the k-Shortest Paths. SIAM Journal on Computing, Vol. 28, 1999, No. 2, pp. 652–673, doi: 10.1137/S0097539795290477.

[7] Erjavec, T.: MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC '04), ELRA, 2004, pp. 1535–1538.

[8] Erjavec, T.: Multilingual Tokenisation, Tagging, and Lemmatisation with Totale. Proceedings of the 9th INTEX/NOOJ Conference, 2006, pp. 33–34.

[9] Erjavec, T.: MULTEXT-East Version 4: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In: Chair, N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D. (Eds.): Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC '10), Valletta, Malta, ELRA, 2010, pp. 2544–2547.

[10] Hajič, J.: Morphological Tagging: Data vs. Dictionaries. Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (NAACL 2000), 2000, pp. 94–101.

[11] Hajič, J.—Hric, J.—Kuboň, V.: Česílko – An MT System for Closely Related Languages. Proceedings of ACL 2000, 2000, pp. 7–8, doi: 10.3115/974147.974149.

[12] Hajič, J.—Homola, P.—Kuboň, V.: A Simple Multilingual Machine Translation System. In: Hovy, E., Macklovitch, E. (Eds.): Proceedings of the MT Summit IX, New Orleans, USA, AMTA, 2003, pp. 157–164.

[13] Homola, P.—Kuboň, V.: Improving Machine Translation Between Closely Related Romance Languages. Proceedings of EAMT, 2008, pp. 72–77.

[14] Vičič, J.—Homola, P.—Kuboň, V.: Automated Implementation Process of a Machine Translation System for Related Languages. Computing and Informatics, Vol. 35, 2016, No. 2, pp. 441–469.

[15] Karlsson, F.—Voutilainen, A.—Heikkilä, J.—Anttila, A. (Eds.): Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin, New York, 1995.

[16] Levenshtein, V. I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Doklady Akademii Nauk SSSR, Vol. 163, 1965, No. 4, pp. 845–848 (in Russian).

[17] Scannell, K. P.: Machine Translation for Closely Related Language Pairs. Proceedings of the Workshop Strategies for Developing Machine Translation for Minority Languages, Genoa, Italy, 2006, pp. 103–109.

[18] Sheikh, Z. M. A. W.—Sánchez-Martínez, F.: A Trigram Part-of-Speech Tagger for the Apertium Free/Open-Source Machine Translation Platform. Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation, Alicante, 2009, pp. 67–74.

[19] Snover, M.—Dorr, B.—Schwartz, R.—Micciulla, L.—Makhoul, J.: A Study of Translation Edit Rate with Targeted Human Annotation. Proceedings of AMTA Conference, 2006, pp. 223–231.

[20] Tyers, F. M.—Donnelly, K.: Apertium-cy – A Collaboratively-Developed Free {RBMT} System for Welsh to English. The Prague Bulletin of Mathematical Linguistics, Vol. 91, 2009, pp. 57–66, doi: 10.2478/v10108-009-0016-4.

[21] Tyers, F. M.—Wiechetek, L.—Trosterud, T.: Developing Prototypes for Machine Translation Between Two Sámi Languages. Proceedings of 13th Annual Conference of the EAMT, 2009, pp. 120–127.

[22] Vičič, J.: Rapid Development of Data for Shallow Transfer RBMT Translation Systems for Highly Inflective Languages. Proceedings of the Conference on Language Technologies, Institut Jožef Stefan, Ljubljana, 2008, pp. 98–103.

[23] Vičič, J.—Forcada, M. L.: Comparing Greedy and Optimal Coverage Strategies for Shallow-Transfer Machine Translation. Proceedings of the International Conference on Intelligent Information Systems XVI (IIS '08), 2008, pp. 307–316.

[24] Vičič, J.—Homola, P.: Speeding Up the Implementation Process of a Shallow Transfer Machine Translation System. Proceedings of the 14th EAMT Conference, Saint Raphael, France, 2010, pp. 261–268.

[25] Vičič, J.—Homola, P.—Kuboň, V.: A Method to Restrict the Blow-Up of Hypotheses of a Non-Disambiguated Shallow Machine Translation System. Proceedings of the International Conference RANLP, Borovec, Bulgaria, ACL, 2009, pp. 460–464.

[26] Vičič, J.—Kuboň, V.—Homola, P.: Česílko Goes Open-Source. The Prague Bulletin of Mathematical Linguistics, Vol. 107, 2017, pp. 57–66, doi: 10.1515/pralin-2017-0004.

[27] Welch, L. R.: Hidden Markov Models and the Baum-Welch Algorithm. IEEE Information Theory Society Newsletter, Vol. 53, 2003, No. 4, 5 pp.

Jernej Vičič is Assistant Professor at the University of Primorska, Primorska Institute for Natural Sciences and Technology in Koper, Slovenia. His main research interest is in the field of hybrid machine translation for related languages.

Marko Grgurovič is Teaching Assistant at the University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies in Koper, Slovenia.