Machine translation system modeling based on sentences comparison

Alexander Sak
Moscow State University of Civil Engineering, Yaroslavskoe shosse, 26, Moscow, 129337, Russia
sakan@mgsu.ru

Abstract: The problem urgency is determined by an increasing need for natural language processing systems especially in the area of construction and civil engineering due to an enormous amount of document in foreign languages to deal with. Author’s purpose is to figure out an approach to machine translation system based on initial sentence and online dictionaries examples comparison. Although online dictionaries are designed to be used by humans, application of some principles based on the theory of graphs and syntactic analysis as well as methods of sets allows to use them as thesaurus for natural processing systems. The author proves the reliability of this approach by using the properties of a relation such as reflexivity, symmetry and transitivity. As a result the author displays a part of a syntactic analysis block of his computer program as well as a block of the program responsible for comparing sentences on the basis of fundamental products. At the end of the paper the author comes to the conclusion that such an approach can be extended not only to the translation problem but also to a broad range of natural language processing issues.

1. Introduction

When modeling a machine translation system, one of the key tasks is to obtain the correct translation of the predicate [1]. If a predicate, represented in English as a rule by a verb, is translated correctly, then this will allow to fill in correctly the valencies of the verb and adequately express the meaning of the sentence in Russian. In the beginning of the machine translation epoch many language-processing systems were designed by developing a set of hand-written rules [8], [9] such as by writing grammars or devising heuristic rules for stemming. Since the "statistical revolution" in the late 1980s and mid-1990s, much natural language processing research was carried out on machine learning [10], [11]. The machine-learning approach is focused on learning automatically certain rules through the analysis of large corpora of typical real-world examples rather than on using statistical inference. Example-based machine translation (EBMT) approach was proposed by Makoto Nagao in 1984 [12]. It is based on the idea of analogy. In this approach, the corpus that is used contains texts that have already been translated. Given an input sentence, sentences containing similar sub-sentential components are selected from this corpus. The similar sentences are then used to translate the sub-sentential components of the original sentence into the target language, and these phrases are put together to form a complete translation. This approach seems to be the most suitable as far as texts specialized in construction are concerned. On the other hand the corpora used by EBMT have to be thoroughly prepared for such translations without deep linguistic analysis. In this paper we introduce some new
mathematical methods to deal with example-based approach. When using online dictionaries as
thesauri which are structured for a human usage such methods as syntactic analysis and sentence
components transforming into fundamental products prevent the system from drowning under rules-
based heavy-weight structures. This approach puts the emphasis on programming methods handling
online dictionaries entries as well as discrete mathematical methods dealing with comparative
analysis.

2. Methods

When we develop a model of a machine translation system based on a comparison of the original
sentence and the examples included in the online dictionary, the main issue is the equivalence relation
between these sentences. The established equivalence relation between the original sentence and the
dictionary example will allow to come up with an idea of the structure of this phrase in Russian and
find the translation of the verb depending on its valencies. Let us imagine the predicate sentence group
as a nonempty set $A = \{ \text{put, out, a fire} \}$, which appears in the online dictionary, and look at the
nonempty set $B = \{ \text{put, out, a candle} \}$, which occurs in the original sentence.

2.1. Equivalence relation.

First of all, it is necessary to prove the equivalence relation on the set of sentences with similar lexical
and grammatical characteristics. The relation $R$ over the sets $A$ and $B$ is called the equivalence relation
if $R$ over the corresponding elements of $A$ and $B$ is reflective, symmetric and transitive [6].

1. For each $a \in A, b \in B$, $(a, a) \in R$

For example, for the sets $A$ and $B$ determined by the verb to put, the combination of $(\text{put U out}) \in A$
and $(\text{put U out}) \in B$ will have the same meaning due to the reflexivity of a (out).

2. If $(a, b) \in R$, then $(b, a) \in R$.

Let’s imagine that $a \in A, b \in B$, and $R$ means the genus-species relation is Part of, which is often
used in semantic networks. If $a$ and $b$ are types of some generic concept, then they are related by the
relation $R$ and, accordingly, $(a, b) \in R$, then $(b, a) \in R$, i.e. they are symmetrical and retain the same
meaning of expression in both sentences. For example, there is a candle R a lantern, where the generic
word (R) is source of light. If the meaning of the expression requires the condition $a = b$, in an idiom,
for example, from the horse’s mouth, such a relation is called partially ordered and antisymmetric. If
you replace the word horse with another word, the meaning of the expression will not be the same.
Despite this, antisymmetry does not affect the equivalence of sets, and the expressions remain
equivalent, because reflect the same meaning.

3. If $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$.

The relation $R$ over the sets $A$ and $B$ is transitive if, for each pair $(a, b) \in R$ and $(b, c) \in R$, i.e. there
are $aRb, dRc$. The relation $R$ is not transitive if there exist elements $a, b, c \in A$ such that $(a, b) \in R$
and $(b, c) \in R$, but the pair $(a, c) \not\in R$. In case of comparing the original sentence and the example
from the dictionary, if $a$ and $b$ have the same meaning as $b$ and $c$, where $c$ is the meaning of the
sentence translated into Russian, then the equivalence relation will remain between $a$ and $c$. This can
be conveyed by the example of the composition of relations [6].
Figure 1. An arrow diagram of sense transition from an English expression to the Russian one going through an online dictionary example of this expression.

It is not necessary to represent sentences as separate sets. If two sentences are represented as subsets of the same set $S = \{\text{put, down, a fire, a candle}\}$, then the relation $R$ on $S$ will also be an equivalence relation, then the same properties of reflexivity, symmetry and transitivity will correspond to the equivalence relation over a given set of lexemes.

2.2. Syntactic analysis.
In order to combine the elements of a sentence into a set, it is necessary to carry out a syntactic analysis or parsing. The syntactic analysis was developed by Chomsky [3], however, it remains relevant until now. For example, let’s consider the sentence *I read a book at the university*.

![Syntactic Analysis Tree]

Figure 2. An example of a syntactic analysis performed automatically.

2.3. Fundamental products.
*To read* is a verb that has many meanings, for example *read about* (to read information concerning someone), *read as* (to interpret someone or something as something), *read something back* (to read back some information to the person who has just given it), *read between the lines* (to understand more than is written or stated), *to read from, to read for, to read of, to read to, to read up to*, etc. Each set of prepositions and direct complements (in accordance with their affiliation to classes) with the verb *to read* will have a meaning different from the meaning of another set of prepositions and objects.
with the verb (i.e., another set). It is clear that the direct complements in the original phrase and those in the example will be different. Comparing them in terms of belonging to one class of words can be a solution to this problem. To take an example: string[] keywords = { "state", "action", "activity", "process", "weight", "form", "variety", "event", "information", "composition", "piece", "exercise", "person", "task", "occupation", "position", "duty", "responsibility", "affair", "matter", "occurrence", "state of affairs", "requirement", "detail", "thing", "purpose", "scheme", "trick", "pattern", "effect", "reaction", "operation", "instrument", "machine", "tool", "vehicle", "body", "show", "area" };

If a direct complement in the original sentence belongs to one or several of these classes, and the corresponding lexeme of the example from the dictionary belongs to the same class or classes, then we can assume that they belong to the same class which significantly increases the chances of the meaning coincidence in relevant sentences. How to formalize the corresponding elements of the verb environment for comparison? The set algebra and full normal forms will come to our rescue. In general, the meaning of an expression and a sentence as a whole can be regarded as a fundamental product if it consists of \( n \) literals with \( n \) variables.

There is an expression “put it away”. Let’s represent it as a product: \( (\text{put} \cap \text{away} \cap \text{it}) \).

The explanatory dictionary contains the following products with explanations: \( (\text{put} \cap \text{away} \cap \text{something}) = 1) \) put something into the proper place; 2) ship-move away; \( (\text{put} \cap \text{away} \cap \text{a person}) = 1) \) get rid of someone; 2) put someone in prison or asylum; \( (\text{put} \cap \text{away} \cap \text{some ideas}) = \text{leave behind}; [2] \) Therefore, it is necessary to bring the variable \( \text{it} \) to accordance with the dictionary meanings. Using it as an abstract variable that does not imply \( \text{it} - \text{something} \) relation, we must embrace all the elements of a given set. Let’s represent this variable as a union \( (\text{C} \cup \text{C}^c) \), i.e. \( (\text{a thing} U \text{a person}) \), thus we obtain the product \( E = (\text{put} \cap \text{away} \cap \text{a thing}) U (\text{put} \cap \text{away} \cap \text{a person}) \), that is to say, this work covers all vocabulary meanings, which is good. There are no exceptions left uncovered. It is usually a demonstrative pronoun indicating an inanimate object, an animal, an event or fact. Thus, our expression is \( E = (\text{put} \cap \text{away} \cap \text{a thing}) \) will match the expression \( E_1 = (\text{put} \cap \text{away} \cap \text{something}). \)

![Figure 3. Representation of a lexical construction by means of an intersection of sets.](image-url)
Hence, one concept corresponds to the set, and the other one corresponds to the complement of the set, that is, one can figure out this union of intersections as follows:

\[
\text{Figure 4. Intersection of a set and its complement}
\]

Thus, when comparing sentences, our goal is to determine the most relevant elements using semantic analysis and to convert the key elements of the linguistic environment of the original sentence predicate into literals for their further transformation into a fundamental product.

We can assign the literal \( A \) to the direct complement, assign \( B \) to the preposition, and \( C \) to the indirect complement. For example, for the next sentence \( I \ read \ an \ interesting \ story \ in \ a \ thick \ book \), we can define the following product \( E = story \cap in \cap book = A \cap B \cap C \). Looking through the examples in the dictionary, we compare the corresponding elements, and if they correspond to the literals of the original sentence, we assign them the value of a variable, if they do not match, we assign them the value of the variable complement. We compare nouns in terms of their belonging to the class, and prepositions in terms of their coincidences. We have examples as follows:

- I read you as a quiet guy \( E_1 = A \cap B \cap C \);
- I can read between the lines \( E_2 = A \cap B \cap C \);
- Please read this manuscript for spelling errors \( E_3 = A \cap B \cap C \);
- I read an interesting article in today’s newspaper \( E_4 = A \cap B \cap C \);
- I think I have read of you in the papers \( E_5 = A \cap B \cap C \);

We have a partial coincidence of the literal values in \( E_2, E_3 \) and \( E_5 \) and a complete coincidence in \( E_4 \), so \( E_4 \) will be the key to the translation.

3. Results

Here is an example of some blocks of code for parsing a narrative sentence.

The following listing provides a part of the syntactic analysis unit responsible for processing the attributive syntax relation.

```java
if (blant.Value.GetRel().IndexOf("Det") != -1)
{
    k = 0;
    str100 = blant.Value.GetRel();
    str100 = str100.Replace("Det", "");
    Point10 = new Point("Det" + str100, null);
    s = 0;ss = 0;
    for (int f = 9; f >= 0; f--)
    {
        if (blant.Value.GetPoint(f) != null)
        {
            s++;
        }
    }
}
```
if (k == 1)
{ structure1.AddFirst(new matrix(blant.Value.GetPoint(f).GetName(),
blant.Value.GetPoint(f).GetWord(), blant.Value.GetPoint(f).GetNode())); ss++; }
if (blant.Value.GetPoint(f).GetNode().GetAdjective(0) != null || blant.Value.GetPoint(f).GetName() == "article" || blant.Value.GetPoint(f).GetName() == "adjective")

This block processes articles, adjectives, and participles determined by nouns.

if (k == 0)
{
if (blant.Value.GetPoint(f).GetName() != "article")
Point2 = new Point("adjective", new matrix("adjective", blant.Value.GetPoint(f).GetWord(),
blant.Value.GetPoint(f).GetNode()));
else
{ Point2 = new Point("article", new matrix("article", blant.Value.GetPoint(f).GetWord(), null)); k = 1; }
rlinq.AddLast(new bigraph(Point10, Point2));
}

if (blant.Value.GetPoint(f).GetNode().GetAdverb(0) != null ||
blant.Value.GetPoint(f).GetNode().GetNoun(0) != null)
{ k = 1; ss++; }

if (structure1.Count() > 0 && ss != s)
{ mixture.AddLast(new list("Circ" + circ.ToString(), structure1));
structure1.Clear();
rlinq.AddLast(new bigraph(new Point(blant.Value.GetRel(),null), new Point("Circ" +
circ.ToString(), null)));
circ++; } 
k = 0;
mixture.Remove(blant);
}

The next block of the parsing program is responsible for the circumstantial syntactic relations.

if (blant.Value.GetRel().IndexOf("Circ") != -1)
{
k = 0;
for(int i = 9;i>=0;i--)
{ if (blant.Value.GetPoint(i) != null)
{ if (k > 0)
structure.AddLast(new Form1.matrix(blant.Value.GetPoint(i).GetName(),
blant.Value.GetPoint(i).GetWord(), blant.Value.GetPoint(i).GetNode()));
if (blant.Value.GetPoint(i).GetNode().GetAdverb(0) != null &&
blant.Value.GetPoint(i).GetNode().GetAdverb(1).Length > 5 &&
blant.Value.GetPoint(i).GetNode().GetAdverb(2).Length > 5)
When processing data as a result of the specific way of keeping the content of data in online
dictionaries, it is necessary to consider each case separately

```
k = 1;
else
    if (blant.Value.GetPoint(i).GetNode().GetNoun(0) != null &&
        blant.Value.GetPoint(i).GetNode().GetNoun(0).Length > 10 &&
        blant.Value.GetPoint(i).GetNode().GetNoun(1).Length > 10)
        k = 2;
    if (k == 1)
        { rlinq.AddLast(new bigraph(new Point(blant.Value.GetRel(), null), new
            Point("adverb", new matrix("adverb", blant.Value.GetPoint(i).GetWord(),
            blant.Value.GetPoint(i).GetNode()))));
    if (k == 2)
        { rlinq.AddLast(new bigraph(new Point(blant.Value.GetRel(), null), new
            Point("noun", new matrix("noun", blant.Value.GetPoint(i).GetWord(),
            blant.Value.GetPoint(i).GetNode()))));
```

Here is a part of program responsible for transforming parsed elements of a sentence into literals and
fundamental products. The advantage of this approach is that it allows to resort even to semantics of
respective sentences by checking out their components belonging to a similar class.

```
while (current1 != null)
```

```
A = false; B = false; C = false; D = false;
if (node1.Value.GetElement(0).GetPart() == "noun")
```

```
if (current1.Value.GetElement(0).GetPart() == "noun")
```

```
for (int i = 0; i < 10; i++)
```

```
for (int j0 = 0; j0 < 10; j0++)
```

```
if (node1.Value.GetElement(0).GetNounFeat(i) != null &&
    node1.Value.GetElement(0).GetNounFeat(i) == current1.Value.GetElement(0).GetNounFeat(j0))
    A = true;
```
This compares the class attributes of the literals of the original sentence `node1.Value.GetElement(0).GetNounFeat(i)` to the nouns of the example `current1.Value.GetElement(0).GetNounFeat(j0)` and if they match, A is assigned the value of true

```csharp
    if (node1.Value.GetElement(i).GetPart() == "preposition")
    {
        if (current1.Value.GetElement(j12) != null)
            if (current1.Value.GetElement(j12).GetPart() == "preposition")
                if (node1.Value.GetElement(i).GetWord() == current1.Value.GetElement(j12).GetWord())
                    if (j1 == 0) { A = true; j1++; }
                    else if (j1 == 1) { B = true; j1++; }
                    else if (j1 == 2) { C = true; j1++; }
                    else if (j1 == 3) { D = true; j1++; }
    }

If the prepositions in the original sentence and an example match, the corresponding literal is assigned the value of true

```csharp
    if (i < node1.Value.GetNumber() - 1 && j12 < current1.Value.GetNumber() - 1)
        if (node1.Value.GetElement(i + 1).GetPart() == current1.Value.GetElement(j12 + 1).GetPart())
            if (node1.Value.GetElement(i + 1).GetPart() == "noun")
                for (int x = 0; x < 20; x++)
                    for (int y = 0; y < 20; y++)
                        if (node1.Value.GetElement(i + 1).GetNounFeat(x) != null && node1.Value.GetElement(i + 1).GetNounFeat(x) == current1.Value.GetElement(j12 + 1).GetNounFeat(y))
                            if (j1 == 0) { B = true; j1++; }
                            else if (j1 == 1) { C = true; j1++; }
                            else if (j1 == 2) { D = true; j1++; }

And if the prepositions coincide and the nouns following them also coincide as parts of the same class, this indicates a very large degree of coincidence of the two sentences.

```csharp
    if (current1.Value.GetElement(0) != null)
        for (int i = 0; i < 10; i++)
            newnouns[i] = current1.Value.GetElement(0).GetNounFeat(i);
        sak1.AddLast(new sentence(current1.Value.GetElement(0).GetWord(),
            current1.Value.GetElement(0).GetPart(), "A", A, newnouns));
```

After determining the values of literals, their values are written down into the collection for subsequent comparison and analysis of the results.
if (current1.Value.GetElement(1) != null)
{
    for (int i = 0; i < 10; i++)
        newnouns[i] = current1.Value.GetElement(1).GetNounFeat(i);
    sak1.AddLast(new sentence(current1.Value.GetElement(1).GetWord(),
                       current1.Value.GetElement(1).GetPart(), "B", B, newnouns));
}
if (current1.Value.GetElement(2) != null)
{
    for (int i = 0; i < 10; i++)
        newnouns[i] = current1.Value.GetElement(2).GetNounFeat(i);
    sak1.AddLast(new sentence(current1.Value.GetElement(2).GetWord(),
                              current1.Value.GetElement(1).GetPart(), "B", B, newnouns));
}
if (current1.Value.GetElement(3) != null)
{
    for (int i = 0; i < 10; i++)
        newnouns[i] = current1.Value.GetElement(3).GetNounFeat(i);
    sak1.AddLast(new sentence(current1.Value.GetElement(3).GetWord(),
                              current1.Value.GetElement(1).GetPart(), "B", B, newnouns));
}
hello1.AddLast(new text(sak1, current1.Value.GetRussian(), sak1.Count()));
sak1.Clear();
current1 = current1.Next;            }

4. Discussion

By representing a sentence as a binary graph the author manages to find the linguistic environment of a predicate that later is transformed into a set. That analysis is based on linguistic information extracted from common online dictionaries entries rather than on statistics like those ones used in some EBMT systems. The equivalence of sentences to compare is proven by displaying such properties of relations as reflexive symmetric and transitive closure of a relation. The sets comprised of direct and indirect complements and prepositions represented as fundamental products can be easily compared to each other in order to find maximum coincidence and proceed to its proper translation. At the next stage of our research dealing with word combinations translation we’ll combine linguistic information methods with self-trained method learning from statistics over the data.

5. Conclusions

Given approach of using graphs and sets to find the predicates and relevant elements of their environments to transform them into fundamental products seems to be more reliable that one based only on statistical analysis. Online dictionaries corpora contain enough examples to make high-level translation. Specialized online dictionaries will allow to cope with translations from civil engineering area. Such an approach can be extended not only to the translation problem but to a broad range of natural language processing issues. This approach can be applied to machine learning and neural networks when the block of Russian sentence analysis is developed.
References

[1] Sak A.N. 2015 Identification of the sentence members as the main task of machine translation (Moscow: Scientific Review Ed.14).

[2] Alyoshintsev A.V. and Sak A.N. 2019 Analysis of natural language sentences by methods of the theory of graphs and the theory of sets (Moscow: Proceedings of international conference FRUCT) p 556

[3] Chomskiy N. and Miller Dj. 2003 Introduction to natural language analysis (Moscow: URSS)

[4] Steven S. 2011 Skiena Algoritm. The algorithm Design Manual (St. Petersburg: BHV).

[5] Paducheva E. V. 2009 Articles of different periods (Moscow: Studia philologica, p 20).

[6] Kazanskiy A.A. 2008 Discrete mathematics (Moscow: Publisher Balabanov I.V.) p 78.

[7] Alyoshintsev A. V., Bessonova E.V., and Sak A. N. 2017 Modeling of the machine translation system for technical texts of the telecommunication field using the object-oriented programming language C# (Moscow: T-Comm, Telecommunications and transport Vol 11. №10) pp. 66-73

[8] Winograd, Terry 1971 Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. (http://hci.stanford.edu/winograd/shrdlu/)

[9] Roger C. Schank and Robert P. Abelson 1977 Scripts, plans, goals, and understanding: An inquiry into human knowledge structures (Psychology Press; 1 edition) p 150

[10] Philip Resnik 2011 Four revolutions (Language Log, February 5).

[11] Mark Johnson 2009 How the statistical revolution changes (computational) linguistics (Proceedings of the EACL Workshop on the Interaction between Linguistics and Computational Linguistics).

[12] Makoto Nagao 1984 A framework of a mechanical translation between Japanese and English by analogy principle.(Elsevier Science Publishers. B.V. A. Elithorn and R. Banerji editors.).