УДК 811.111:69

АВТОМАТИЧЕСКИЙ АНАЛИЗ ГРАММАТИКИ АНГЛИЙСКОГО ПРЕДЛОЖЕНИЯ ПРИ РЕАЛИЗАЦИИ МАШИННОГО ПЕРЕВОДА ТЕКСТОВ ПО СТРОИТЕЛЬНОЙ ТЕМАТИКЕ

Е.В. Бессонова, А.Н. Сак

*Национальный исследовательский Московский государственный строительный университет (НИУ МГСУ),
129337, г. Москва, Ярославское шоссе, д. 26*

Аннотация. Необходимость в постоянном доступе к научным статьям или инструкциям на английском языке по применению строительной техники и устройств, а также обработка информации о новых технологиях, требуют разработки системы машинного перевода для строительной отрасли. Развитие языков объектно-ориентированного программирования позволяет представлять объекты, содержащие большое количество информации, наиболее удобным способом с точки зрения стоящих перед разработчиком задач. В статье рассматриваются вопросы представления лексических единиц предложения как объектов программы на языке C#, а также вопросы особенностей перевода соответствующих лексем, учитывая грамматику английского и русского языков в области строительства. Авторы пытаются избежать представления лексико-семантической структуры предложения посредством метаязыка — посредника, значительно утяжеляющего систему перевода, путем создания более легкой и понятной структуры. Для этого устанавливаются определенные правила лексического и грамматического соотношения повествовательного предложения на английском и русском языках. Учитываются особенности обработки html-страниц онлайн словарей, используемых как тезаурус лексических единиц при переводе.

Ключевые слова: машинный перевод, объектно-ориентированное программирование, лексема, грамматика, массивы, коллекции, онлайн словари

DOI: 10.22227/1997-0935.2017.1.107-116

AUTOMATED ANALYSIS OF AN ENGLISH SENTENCE GRAMMAR WHILE EXECUTING A COMPUTER TRANSLATION OF TEXTS ON CIVIL ENGINEERING

E.V. Bessonova, A.N. Sak

*Moscow State University of Civil Engineering (National Research University) (MGSU),
26 Yaroslavskoe shosse, Moscow, 129337, Russian Federation*

Abstract. The need for permanent access to scientific articles in English, or instructions on the use of construction equipment and devices, as well as the need for processing an information about new technologies mean that the development of a computer translation system for the construction industry is necessary. Development of object-oriented programming languages allows to present objects containing a great deal of information in the most convenient way in terms of tasks facing a developer. The article deals with problems of presentation of sentence lexical units as objects in C# language. Issues of specific features of translation of corresponding lexemes in view of English and Russian grammar in the sphere of construction are also regarded in the article. Authors tried to avoid presenting a lexical-semantic structure of sentence by means of metalanguage which is a mediator making the translation system considerably heavier, by creating a more easy and understandable structure. For this purpose, certain rules of lexical and grammatical correlation of a declarative sentence in the English and Russian languages are established. Specific features of processing of html-pages of online dictionaries used as thesauri of lexical units in translation are taken into account. Development of automation means for translation in the sphere of construction is beneficial even for small amounts of a text being translated because in this case the human factor is excluded.

Key words: computer translation, object-oriented programming, lexeme, grammar, arrays, collections, online dictionaries

Проблема машинного перевода находится на стыке лингвистики и кибернетики и является одной из задач, стоящих при изучении проблем искусственного интеллекта. Систему машинного перевода для строительной сферы можно рассматривать как информационную систему, способствующую улучшению организации логистики строительства.

Объектно-ориентированные языки программирования, например C#, требуют от создателей систем машинного перевода нахождения новых подходов к реализации данных систем исходя из возможностей этих языков и поиск тех лингвистических методов, которые сочетаются с возможностями современного программирования.

На первом этапе анализа осуществляется запрос о каждой лексической единице предложения к находящемуся онлайн толковому словарю. Важно отметить, что использование онлайн-словарей узкой специализации в области строительства позволяет достичь наиболее адекватного перевода в каждом из строительных направлений. Возможности употребления слов в качестве разных частей речи отражаются в соответствующих коллекциях, входящих в объект класса, характеризующего данную лексическую единицу. Обладая коллекциями употреблений и свойств каждой лексемы предложения, мы можем идентифицировать каждое слово с точки зрения его принадлежности к той или иной части речи, что очень важно при переводе, так как для решения проблемы омонимии (многозначности) необходимо определить, к какой части речи относится лексема в данном контексте [1].

Когда слова предложения распределены по частям речи, т.е. можно сказать, что завершен процесс анализа предложения в нашем исследовании, необходимо переходить к адаптации полученных лексем к русской грамматике. Сравнивая повествовательные (как наиболее часто встречающиеся при переводе научных и, в частности, строительных текстов) предложения русского и английского языков, можно сказать, что в принципе грамматическая структура их схожа: подлежащее–сказуемое–прямое дополнение–косвенное дополнение–обстоятельство места–обстоятельство времени. Так же, как и в английском языке, порядок членов предложения в русском повествовательном предложении может варьироваться в рамках определенных правил, что дает достаточное пространство для маневра [2]. Основным отличием выступает тот факт, что в русском языке склоняются существительные, прилагательные и причастия, в английском языке отношения между членами предложения осуществляются в основном за счет предлогов [3]. Английские прилагательные стоят перед существительными, как и в русском языке, просто в качестве прилагательных в английском часто используются существительные: *real estate agency* — агентство недвижимости. Слово *real estate* — суще-

ствительное, если и удастся связать его с понятием «жилье» при переводе можно будет «заменить» его и на производное от этого слова прилагательное «жилищный». Если же такого прилагательного нет, можно поставить существительное в значении определения после определяемого существительного в родительном падеже: агентство недвижимости.

Необходимо также рассказать о технологии формирования и передачи характеристик для обработки уже полученных в результате перевода слов. В матрице *sentencemembers* содержится подробная информация о каждом слове предложения. В 0-ой строке — само английское слово, в 1-ой строке — часть речи, которой оно соответствует. Если это глагол, то в 2-ой строке указывается в зависимости от обнаруженного сказуемого, в каком числе (единственном или множественном) стоит форма глагола и какому лицу соответствует его форма. Если речь идет о существительном, тогда во 2-ой строке будет указано в единственном или множественном числе оно употребляется. Смысл процесса синтеза заключается в том, чтобы по лингвистическому окружению решить, в какой форме будет употребляться соответствующее слово или конструкция на языке перевода.

Если, например, вместе идут два имени существительных, между которыми стоит предлог «of», значит, скорее всего, между словами существует отношение принадлежности и на русском второе существительное нужно ставить в родительном падеже, а про предлог естественно забыть. Условлено не переводить предлоги при помощи словаря, а предусмотреть способ перевода в программе. В данном случае надо убрать предлог и поставить следующее за предлогом существительное в родительном падеже. Разбор идет последовательно, т.е. переводится слово за словом, и формируется матрица *sentencemembers1*, куда будут помещаться русские слова после перевода, а их характеристики будут заранее определяться за счет лингвистического окружения. Поэтому характеристика слова — то, что оно должно стоять в родительном падеже, будет передаваться уже следующему элементу массива

```
if (ch1 - 1 >= 0 && ch1 + 1 <= super2.Count())
{
    if (sentencemembers[1,ch1-1]=="noun" && sentencemembers[1,ch1+1]=="noun")
        sentencemembers1[3,ch1+1]="genetive";
}
```

Перевод следующего элемента произойдет уже на следующем шаге цикла.

Если встречается комбинация «to» + verb (глагол), то необходимо предлог «to» заменить на «чтобы», напрямую, не обращаясь к словарю. Если же после предлога «to» идет имя существительное, прилагательное или артикль, существует несколько

возможных вариантов перевода. Это может переводиться предлогами «в», «на» или «к» в зависимости от контекста, а если за предлогом следует существительное, оно должно иметь форму дательного падежа на русском. Если речь идет о последовательности предлог «in» + существительное, переводим предлог русским предлогом «в», а существительное ставится в предложный падеж, а если после предлога идет прилагательное, то после перевода прилагательное также должно носить форму предложного падежа. Такие же принципы распространяются и на предлоги «for» — существительное, следующее за ним, будет стоять в винительном падеже, предлогу «from» — соответствует предлог «из» + существительное в родительном падеже, предлогу «about» — соответствует предлог «насчет» и существительное в родительном падеже, предлогу «without» с именем существительным соответствует предлог «без» + существительное в родительном падеже, а если предлог используется с глаголом в форме герундия, надо переводить с помощью деепричастного оборота, например, «не делая что-либо». Предлог «with» должен переводиться при помощи предлога «с» + существительное в творительном падеже.

Принимая во внимание особенности онлайн англо-русского словаря, следует адаптировать указатели частей речи к соответствующим указателям, принятым в словаре: «noun» → «n», «adjective» → «adj», «pronoun» → «pron», если речь идет о наречии («adverb»), то по нашим наблюдениям следует указывать маркер «adj». В данном словаре имеются два маркера части речи для глаголов «vt» и «vi», т.е. транзитивный и не транзитивный глагол или переходный и непереходный. Например, глагол «to run» в нетранзитивном смысле означает «бежать», а в транзитивном, например, «to run an office» означает «управлять офисом». Для того, чтобы выяснить, с каким именно глаголом мы имеем дело, надо проверить, есть ли прямое дополнение у этого глагола.

```

if (sentencemembers[1, ch1] == "verb")
{
    str33 = "vi";
    for (int x1 = ch1; x1 < super2.
Count(); x1++)
    {
        if (sentencemembers[1, x1] == "noun"
|| sentencemembers[1, x1] == "pronoun")
        {
            str33 = "vt"; break;
        }
        if (sentencemembers[1, x1] ==
"preposition")
            break;
    }
}

```

Если на данном этапе цикла встречается фразовый глагол или фразеологическое выражение, необходимо запустить еще один цикл, собирающий все элементы этого устойчивого выражения в единую переменную, которая затем передается на перевод. С точки зрения модели «Смысл ⇔ Текст» здесь наблюдается переход от поверхностно-синтаксической структуры к глубинно-синтаксической [3].

```

if (sentencemembers[1, ch1] == "phrasal verb")
{
    str25 += " ";
    for (int ch2 = ch1 + 1; ch2 < super2.
Count(); ch2++)
    {
        if (sentencemembers[1, ch2] == "phrasal verb")
        { str25 += sentencemembers[0, ch2] + " ";
sentencemembers[0, ch2] = ""; }

        str33 = "<>";
    }
}

```

Так как невозможно приписать фразовому глаголу или фразеологическому выражению соответствующий маркер, который бы воспринимался словарем, переменной присваивается «пустой» параметр str33= "";

Некоторые английские выражения и конструкции, которые прекрасно воспринимает толковый словарь, недоступны для данного англо-русского словаря и приходится производить непосредственную лексическую настройку:

```

if (str25 == "have to " || str25 == "has to ")
    str25 = "have got to";

```

И, наконец, когда информация относительно переводимого слова подготовлена, слово передается на перевод

```

if (sentencemembers[0, ch1] != "the" &&
sentencemembers[0, ch1] != "a" && sentencemembers[0,
ch1] != "" && sentencemembers[1, ch1] != "preposition"
&& sentencemembers[0, ch1] != "be")
{
    inst.translator(ref str25, ref str33);
    rlinq.AddLast(inst.node);
}

```

После обработки слова в классе, отвечающем за перевод, результат передается обратно на форму с помощью ссылки на объект класса, в роли которого выступает коллекция русских значений данного слова [4].

```

sentencemembers1[0, ch1] = inst.
node.GetRussian(0);
}

```

Здесь приводится листинг «грамматического, подготавливающего к переводу» блока программы:

```

string[] prep = { «В», «на», «к» };
int cont = 0; int hp = 0;
translation inst = new translation(); string[,] sentencemembers1 = new string[4, super2.Length];
int ch3 = 0;
string str25 = «»; string str33 = «»; LinkedList<interpreter> rlinq = new LinkedList<interpreter>();

```

С помощью конструктора создается коллекция экземпляров класса interpreter.

```

for (int ch1 = 0; ch1 < super2.Count(); ch1++)
{
    str25 = sentencemembers[0, ch1];
    if (str25 != «»)
    {
        if (sentencemembers[1, ch1] == «noun»)
            str33 = «n»;
        if (sentencemembers[1, ch1] == «pronoun»)
            str33 = «pron»;

        if (sentencemembers[1, ch1] == «verb»)
        {
            str33 = «vi»;
            for (int x1 = ch1; x1 < super2.Count(); x1++)
            {
                if (sentencemembers[1, x1] == «noun» || sentencemembers[1, x1] == «pronoun» || sentencemembe
rs[1,x1]==»adjective»)
                {
                    str33 = «vt»; sentencemembers1[3, x1] = «accusative»;

                    break;
                }
                if (sentencemembers[1, x1] == «preposition»)

                    break;
            }
        }
        if (sentencemembers[1, ch1] == «adjective»)
            str33 = «adj»;

        if (sentencemembers[1, ch1] == «preposition»)
        {
            if (ch1 + 1 < super2.Count())
            {
                ch3 = ch1 + 1; temp0 = sentencemembers[0, ch1] + « «; k = 0;
                while (sentencemembers[1, ch3] == «preposition»)
                {
                    temp0 += sentencemembers[0, ch3] + « «; k = 1;
                    sentencemembers[0, ch3] = null; sentencemembers[1, ch3] = null;
                    ch3++;
                }
            }
            if (k == 1)
            {
                temp0 = temp0.Trim();
                if (temp0 == «instead of»)
                {
                    sentencemembers1[0, ch1] = «ВМЕСТО»; sentencemembers1[1, ch1] = «preposition»;
                    sentencemembers1[2, ch3 + 1] = «genetive»;
                }
            }
        }
    }
}

```

```

        if (temp0 == «in spite of»)
        {
            sentencemembers1[0, ch1] = «несмотря на»; sentencemembers1[1, ch1] = «preposition»;
            sentencemembers1[2, ch3 + 1] = «genetive»;
        }
    }
    else
    {
        if (sentencemembers[0, ch1] == «of»)
        {
            if (ch1 - 1 >= 0 && ch + 1 <= super2.Count())
            {
                if ((sentencemembers[1, ch1 - 1] == «noun» || sentencemembers[1, ch1 - 1] == «pronoun»)
                && sentencemembers[1, ch1 + 1] == «noun» )
                    sentencemembers1[3, ch1 + 1] = «genetive»;
            }
        }
        if (sentencemembers[0, ch1] == «to»)
        {
            if (ch1 < super2.Length)
            {
                if (sentencemembers[1, ch1 + 1] == «verb»)
                { sentencemembers1[0, ch1] = «чтобы»; sentencemembers1[1, ch1] = «preposition»; }
                else
                if (sentencemembers[1, ch1 + 1] == «noun» || sentencemembers[1, ch1 + 1] == «adjective»
                || sentencemembers[1, ch1 + 1] == «article» || sentencemembers[1, ch1 + 1] == «pronoun»)
                {
                    sentencemembers1[0, ch1] = «для»; sentencemembers1[1, ch1] = «preposition»;
                    if (sentencemembers[1, ch1 + 1] == «noun»)
                        sentencemembers1[3, ch1 + 1] = «genetive»;
                    else
                        if (sentencemembers[0, ch1 + 2] != null)
                        {
                            if (sentencemembers[1, ch1 + 2] == «noun»)
                                sentencemembers1[3, ch1 + 2] = «genetive»;
                        }
                }
            }
        }
        if (sentencemembers[0, ch1] == «in»)
        {
            sentencemembers1[0, ch1] = «в»; sentencemembers1[1, ch1] = «preposition»;
            if (ch1 + 1 < super2.Length)
            {
                if (sentencemembers[1, ch1 + 1] == «noun»)
                    sentencemembers1[3, ch1 + 1] = «prepositional»;
                else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «genetive»;
                    }
                }
            }
        }
    }
}

```

```

    }
    if (sentencemembers[0, ch1] == «for»)
    {
sentencemembers1[0, ch1] = «за»; sentencemembers1[1, ch1] = «preposition»;
        if (ch1 + 1 < super2.Length)
        {
            if (sentencemembers[1, ch1 + 1] == «noun»)
                sentencemembers1[3, ch1 + 1] = «acusative»;
            else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «genetive»;
                    }
                }
        }
    }
    if (sentencemembers[0, ch1] == «from»)
    {
sentencemembers1[0, ch1] = «из»; sentencemembers1[1, ch1] = «preposition»;
        if (ch1 + 1 < super2.Length)
        {

            if (sentencemembers[1, ch1 + 1] == «noun»)
                sentencemembers1[3, ch1 + 1] = «genetive»;
            else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «genetive»;
                    }
                }
        }
    }
    if (sentencemembers[0, ch1] == «about»)
    {
sentencemembers1[0, ch1] = «насчет»; sentencemembers1[1, ch1] = «preposition»;
        if (ch1 + 1 < super2.Length)
        {
            if (sentencemembers[1, ch1 + 1] == «noun»)
                sentencemembers1[3, ch1 + 1] = «genetive»;
            else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «genetive»;
                    }
                }
        }
    }
}

```



```

    if (sentencemembers[0, ch1] == «without»)
    {
sentencemembers1[0, ch1] = «без»; sentencemembers1[1, ch1] = «preposition»;
        if (ch1 + 1 < super2.Length)
        {
            if (sentencemembers[1, ch1 + 1] == «noun»)
                sentencemembers1[3, ch1 + 1] = «genetive»;
            else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «genetive»;
                    }
                }
        }
    }
    if (sentencemembers[0, ch1] == «with»)
    {
sentencemembers1[0, ch1] = «с»; sentencemembers1[1, ch1] = «preposition»;
        if (ch1 + 1 < super2.Length)
        {
            if (sentencemembers[1, ch1 + 1] == «noun»)
                sentencemembers1[3, ch1 + 1] = «instr»;
            else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «instr»;
                    }
                }
        }
    }
    if (sentencemembers[0, ch1] == «after»)
    {
sentencemembers1[0, ch1] = «после»; sentencemembers1[1, ch1] = «preposition»;
        if (ch1 + 1 < super2.Length)
        {
            if (sentencemembers[1, ch1 + 1] == «noun»)
                sentencemembers1[3, ch1 + 1] = «genetive»;
            else
                if (ch1 + 2 < super2.Length)
                {
                    if (sentencemembers[0, ch1 + 2] != null)
                    {
                        if (sentencemembers[1, ch1 + 2] == «noun»)
                            sentencemembers1[3, ch1 + 2] = «genetive»;
                    }
                }
        }
    }
}
k = 0;

```

```

    }
    if (sentencemembers[1, ch1] == «adverb»)
        str33 = «adj»;
    if (sentencemembers[1, ch1] == «phrasal verb»)
    {
        str25 += « «; sentencemembers1[1, ch1] = «phrasal verb»;
        for (int ch2 = ch1 + 1; ch2 < super2.Count(); ch2++)
        {
            if (sentencemembers[1, ch2] == «phrasal verb»)
            { str25 += sentencemembers[0, ch2] + « «; sentencemembers[0, ch2] = «»; cont++; }
            str33 = «»;
        }
    }
}

if (sentencemembers[0, ch1] == «that»)
{ sentencemembers1[0, ch1] = «что»; sentencemembers1[1, ch1] = «conjunction»; }
    if (sentencemembers[0, ch1] == «and»)
    { sentencemembers1[0, ch1] = «и»; sentencemembers1[1, ch1] = «conjunction»; }
    if (sentencemembers[0, ch1] == «but»)
    { sentencemembers1[0, ch1] = «но»; sentencemembers1[1, ch1] = «conjunction»; }
    if (str25 == «have to « || str25 == «has to «)
        str25 = «have got to»;
    if (sentencemembers[2, ch1] != «») sentencemembers1[2, ch1] = sentencemembers[2, ch1];

    if (sentencemembers[0, ch1] != «the» && sentencemembers[0, ch1] != «a» && sentencemembers[0,
ch1] != null && sentencemembers[1, ch1] != «preposition» && sentencemembers[0, ch1] != «be» &&
sentencemembers1[0, ch1] != «и» && sentencemembers[0, ch1] != «that»)
    { inst.translator(ref str25, ref str33);
        rling.AddLast(inst.node);
        if (inst.node != null)
            if (inst.node.GetRussian(0) != null)
                Исходя из особенностей англо-русского словаря [5] необходимо учесть определенные особенности при
                обработке html-страницы и стереть определенные элементы.
                trans = inst.node.GetRussian(0);
                trans = trans.Replace(«Перевод», «»);
                if (inst.node.GetRussian(0) != null)
                    if (inst.node.GetRussian(0).Length < 3)
                        trans = trans.Replace(«я «, «»);
                trans = trans.Replace(«история», «»);
                if (sentencemembers[1, ch1] == «phrasal verb»)
                    if (sentencemembers[0, ch1 + cont + 1] != null)
                        if (sentencemembers[1, ch1 + cont + 1] == «preposition»)
                            {
                                string[] words2 = trans.Split(delimiter1);
                                for (int tt = 0; tt < words2.Length; tt++)
                                    for (int zz = 0; zz < prep.Length; zz++)
                                        {
                                            if (words2[tt].Replace(« «, «») == prep[zz])
                                                sentencemembers[0, ch1 + cont + 1] = «»;
                                        }
                            }
                    }
                sentencemembers1[0, ch1] = trans;
                trans = «»
            }
        }
    }
}

```


В данном блоке надо обратить внимание на то, как обрабатываются сложные предлоги типа «in spite of»; «instead of». Каждый из элементов сложных предлогов был внесен в массив предлогов и при определении частей речи был определен как предлог. Затем в приведенном выше блоке запускается цикл, который анализирует простой это предлог или сложный, и если следующее слово также имеет маркер «preposition-предлог», значит, речь идет о сложном предлоге, части которого собираются в единое целое и переводятся. Имеется специальный класс *translation*, непосредственно работающий с англо-русским словарем.

В методе *translator* передаются как аргументы слова на английском языке, предварительно обработанные в двух предыдущих блоках, так и части речи, которым они принадлежат.

Вернув уже обработанную строку, мы продолжаем ее обработку. Затем, если часть речи определена, происходит поиск до первой буквы на русском. Если такая буква найдена, предложение разбивается на слова и слова заносятся в массив. Просматривая его элементы, находится элемент, идентичный обозначению уже определенной части речи, тогда осуществляется переход к следующему за ним элементу. Если этот элемент на русском языке, он заносится в переменную *temp1* и добавляется элемент *temp0* в коллекцию *english*, а если на английском и при условии, что *temp1!=null* значение переменной *temp1* заносится в коллекцию *russian* и добавляется элемент к *temp0*. В случае, если речь идет о фразеологизме или фразовом глаголе, то сначала находится совпадение в строке с выражением на английском, которое было получено в предыдущем грамматическом блоке, а уже потом представляем слова предложения как элементы массива, которые потом заносятся в переменные *temp0* и *temp1*, и отправляем их в коллекции *english* и *russian*. После этого, слово, выраженное данными коллекциями, представляется как экземпляр класса *interpreter*. В данном классе значения объекта *node* на английском в различных выражениях и значения этого же объекта на русском представляются посредством массивов *english1* и *russian1*. Это позволяет координировать английские и русские значения за счет индекса.

В данном блоке был рассмотрен этап предпереводческой подготовки предложения перед осуществлением запроса к двуязычному англо-русскому словарю.

На следующем этапе реализации машинного перевода будет необходимо обработать полученные русские лексемы с точки зрения русской грамматики.

В настоящее время в состав комплексных систем обработки и анализа данных при проектировании в строительстве все чаще включаются подсистемы обработки текстовой информации. Если такие подсистемы предназначены для работы с данными на нескольких языках, перед ними может ставиться задача автоматического перевода с одного языка на другой. Существующие системы машинного перевода совершают ошибки при переводе текстов из строительной области. Например, предложение «My apartment has to be done up before I move in» система машинного перевода Google переводит как «Моя квартира должно быть сделано, прежде чем я двигаться в» вместо «Моя квартира должна быть отремонтирована, прежде чем я туда въеду». Разработка средств автоматизации перевода в строительной области является полезной даже при небольших объемах переводимого текста, так как в этом случае исключается влияние человеческого фактора: устраняется необходимость обращаться к профессиональному переводчику каждый раз, когда инженеру требуется перевод текста, насыщенного незнакомыми ему лексическими и грамматическими единицами. Использование онлайн-словарей соответствующей технической области позволит получить наиболее адекватный перевод. Системы машинного перевода, основанные только на статистических алгоритмах, испытывают большие трудности при переводе фразеологических оборотов и малейшем отклонении от простых грамматических конструкций. Системы, основанные на построении лингвистических правил испытывают трудности с переводом технических терминов. Наша система, использующая онлайн-словари, как двуязычные, так и толковые одноязычные, как общие, так и специализированные, направлена на решение этих противоречий.

ЛИТЕРАТУРА

1. Сак А.Н. Идентификация членов предложения как частей речи как основная задача машинного перевода // Научное обозрение. 2015. Вып. 14. С. 236–240.
2. Мельчук И.А. Язык: от смысла к тексту. М. : Языки славянской культуры, 2012. 174 с.
3. Мельчук И.А. Русский язык в модели «Смысл-Текст». М. : Школа «Языки русской культуры» ; Вена :

Wiener slawistischer Almanach, 1995. 682 с. (Язык. Семиотика. Культура)

4. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual C# в среде разработки Microsoft Visual Studio 2008 и .NET Framework : в 3 чч. М., 2011. Ч. 3: Учебное пособие и практикум. 180 с.

5. Казанский А.А., Сак А.Н. Классы и объекты в языках программирования C# и VBA при решении прикладных задач // Вопросы прикладной математики и вычислительной механики : сб. тр. М. : МГСУ, 2016. № 19. С. 264–274.
6. Казанский А.А., Сак А.Н. Методы искусственного интеллекта при переводе текстов // Вопросы прикладной математики и вычислительной механики : сб. тр. М. : МГСУ, 2015. № 18. С. 248–258.
7. Иомдин Л.Л., Мельчук И.А., Перцов Н.В. Фрагмент модели русского поверхностного синтаксиса //

Научно-техническая информация. Серия 2. 1975. № 7. С. 22–43.

8. Богуславский И.М. Исследования по синтаксической семантике : сферы действия логических слов. М. : Наука, 1985. 175 с.

9. Koehn Philipp. Statistical machine translation. Cambridge : Cambridge University Press, 2010. 433 p.

10. Online dictionary: English-Russian translation of words and expressions, definition, synonyms // Reverso-Dictionary. Режим доступа: <http://dictionary.reverso.net/english-russian/>.

Поступила в редакцию в июне 2016 г.

Об авторах: **Бессонова Елена Владимировна** — кандидат филологических наук, доцент, заведующая кафедрой иностранных языков и профессиональной коммуникации, **Национальный исследовательский Московский государственный строительный университет (НИУ МГСУ)**, 129337, г. Москва, Ярославское шоссе, д. 26, bessonovaev@mgsu.ru;

Сак Александр Николаевич — кандидат филологических наук, доцент кафедры иностранных языков и профессиональной коммуникации, **Национальный исследовательский Московский государственный строительный университет (НИУ МГСУ)**, 129337, г. Москва, Ярославское шоссе, д. 26, sak_inter@mail.ru.

Для цитирования: Бессонова Е.В., Сак А.Н. Автоматический анализ грамматики английского предложения при реализации машинного перевода текстов по строительной тематике // Вестник МГСУ. 2017. Т. 12. Вып. 1 (100). С. 107–116. DOI: 10.22227/1997-0935.2017.1.107-116

REFERENCES

1. Sak A.N. *Identifikatsiya chlenov predlozheniya kak chastey rechi kas osnovnaya zadacha mashinnogo perevoda* [Identification of Sentence Members as Parts of Speech as the Main Task of Machine Translation]. Moscow, Nauchnoye obozrenie Publ., 2015, Issue 14, pp. 236–240. (In Russian)
2. Mel'chuk I.A. *Yazyk : ot smysla k tekstu* [Language : from Meaning to Text]. Moscow, Yazyki Slavyanskoy Kul'tury Publ., 2012, 174 p. (In Russian)
3. Mel'chuk I.A. *Russkiy yazyk v modeli "Smysl-Tekst"* [The Russian Language in the "Meaning-Text" Model]. Moscow, Shkola Yazyki Russkoy Kul'tury Publ., 1995; Vienna : Wiener slawistischer Almanach, 1995, 682 p. (Yazyk. Semiotika. Kul'tura [Language, Semiotics, Culture]). (In Russian)
4. Kazanskiy A.A. *Ob'ektno-orientirovannoe programirovanie na yazyke Microsoft Visual C# v srede razrabotki Microsoft Visual Studio 2008 i .NET Framework : v 3 chch.* [Object-Oriented Programming in Microsoft Visual C# Language in the Microsoft Visual Studio 2008 and .NET Framework Development Environment : in 3 Parts]. Moscow, 2011, Ch. 3 : Uchebnoe posobie i praktikum [Part 3 : Learning Aid and Practicum], 180 p. (In Russian)
5. Kazanskiy A.A., Sak A.N. *Klassy i ob'ekty v yazykakh programmirovaniya C# i VBA pri reshenii prikladnykh zadach* [Classes and Objects in Programming Languages C# and

VBA When Solving the Applied Problems]. *Voprosy prikladnoy matematiki i vychislitel'noy mekhaniki: Sbornik trudov* [Issues of Applied Mathematics and Computational Mechanics: Proceedings]. Moscow, 2016, no. 19, pp. 264–274. (In Russian)

6. Kazanskiy A.A., Sak A.N. *Metody iskusstvennogo intellekta pri perevode tekstov* [Methods of Artificial Intelligence Used for Translation of Texts]. *Voprosy prikladnoy matematiki i vychislitel'noy mekhaniki: Sbornik trudov* [Issues of Applied Mathematics and Computational Mechanics: Proceedings]. Moscow, 2015, no. 18, pp. 248–258. (In Russian)

7. Iomdin L.L., Mel'chuk I.A., Pertsov N.V. *Fragment russkogo poverkhnostnogo sintaksisa* [A Fragment of Russian Superficial Syntax]. *Nauchno-Tekhnicheskaya Informatsiya. Seriya 2* [Science and Technical Information. Series 2]. 1975, no. 7, pp. 22–43. (In Russian)

8. Boguslavskiy I.M. *Issledovaniya po sintaksicheskoy semantike* [Research on Syntactics]. Moscow, Nauka Publ., 1985. 175 p. (In Russian)

9. Koehn Philipp. *Statistical Machine Translation*. Cambridge, Cambridge University Press, 2010, 433 p.

10. Online dictionary: English-Russian translation of words and expressions, definition, synonyms. Reverso-Dictionary. Available at : <http://dictionary.reverso.net/english-russian/>.

About the authors: **Bessonova Elena Vladimirovna** — Candidate of Philological Sciences, Associate Professor, Head of Department of Foreign Languages and Professional Communication, **Moscow State University of Civil Engineering (National Research University) (MGSU)**, 26 Yaroslavskoe shosse, Moscow, 129337, Russian Federation; bessonovaev@mgsu.ru;

Sak Aleksandr Nikolaevich — Candidate of Philological Sciences, Associate Professor, Department of Foreign Languages and Professional Communication, **Moscow State University of Civil Engineering (National Research University) (MGSU)**, 26 Yaroslavskoe shosse, Moscow, 129337, Russian Federation; sak_inter@mail.ru.

For citation: Bessonova E.V., Sak A.N. Avtomaticheskii analiz grammatiki angliyskogo predlozheniya pri realizatsii mashinnogo perevoda tekstov po stroitel'noy tematike [Automated Analysis of an English Sentence Grammar while Executing a Computer Translation of Texts on Civil Engineering]. *Vestnik MGSU* [Proceedings of Moscow State University of Civil Engineering]. 2017, vol. 12, issue 1 (100), pp. 107–116. (In Russian) DOI: 10.22227/1997-0935.2017.1.107-116