ОПЫТ ЛАТЫШСКО↔РУССКОГО МАШИННОГО ПЕРЕВОДА

LATVIAN↔RUSSIAN MACHINE TRANSLATION EXPERIENCE

Горностай Т. *(tatjana.gornostaja@tilde.lv),* ***Васильев А.*** *(andrejs@tilde.lv)*
Скадиньш Р. *(raivis.skadins@tilde.lv),* ***Скадиня И.*** *(inguna.skadina@tilde.lv),*
компания Tilde, Рига (www.tilde.com)

В докладе представлена пилотная версия многоязычного словаря с элементами машинного перевода. Описаны основные этапы обработки текста и архитектура словаря. Обозначены лингвистические трудности латышско↔русского перевода.

1. Введение

На постсоветском пространстве уже более 15 лет ведутся исследования в области автоматической обработки (АО) национальных языков, в том числе и латышского. Латвия входит в состав ЕС, а также является страной с почти половиной русскоязычного населения. В данной ситуации возникает необходимость в программном продукте (ПП), который предоставит пользователю инновационные языковые технологии для решения проблем, связанных с огромным информационным потоком, с одной стороны, и адаптацией носителей русского языка (РЯ) – с другой.

2. История создания

Компания Tilde специализируется в области языковых технологий для балтийских языков.

Начало было положено в 1998 г., когда в свет вышел электронный словарь (ЭС) Tildes Datorvārdnīca, двуязычный латышско↔английский словарь.

В 2000 г. был добавлен латышско↔русский словарь с функциями морфологического анализа, а через два года – латышско↔немецкий словарь.

Параллельно с развитием ЭС шла работа над проектом Viedtulks (интеллектуальный переводчик) [1], [2], [3]. В основу ПП положена технология MoBiMouse – объединение функций ЭС и системы полнотекстового МП [4], [5].

В 2005 г. Viedtulks стал доступным для пользователя. Новый ПП предлагал два режима перевода: пословный перевод и перевод фраз (рисунок 1).

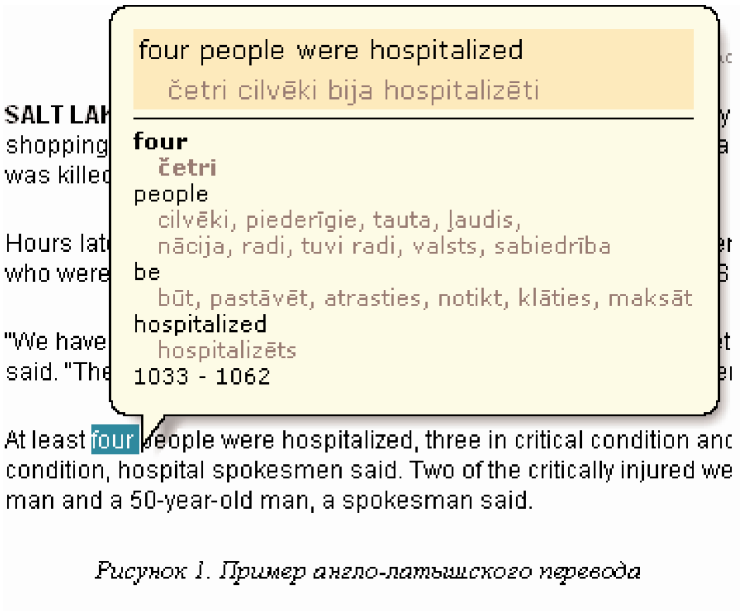

Рисунок 1. Пример англо-латышского перевода

Год спустя была завершена работа над пилотной версией многоязычного словаря с элементами МП Daudzvalodu Vārdnīca. В словарь вошли английский, французский, немецкий, русский, эстонский, литовский и латышский языки.

3. Архитектура словаря

Работа словаря основывается на принципе модульной архитектуры – каждая составляющая отвечает за определенную процедуру. Процесс обработки текста обобщенно отображен на рисунке 2.

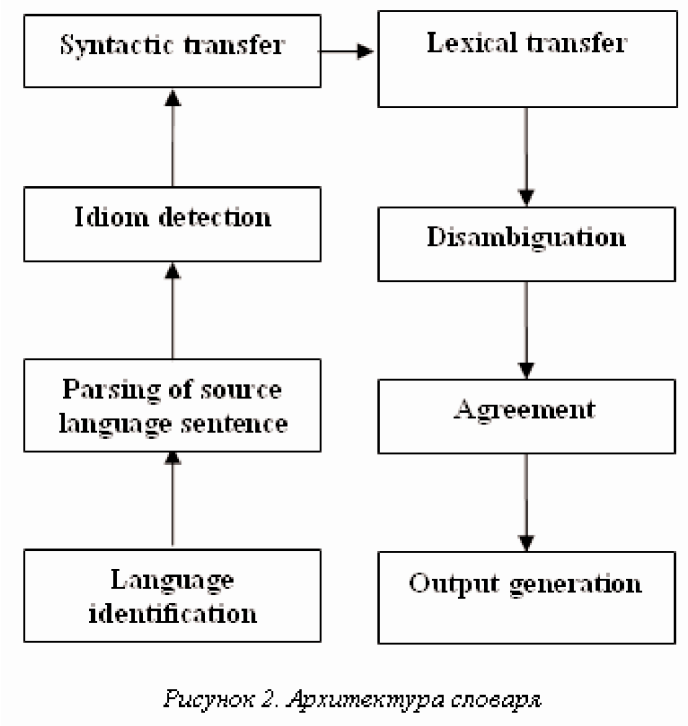

Рисунок 2. Архитектура словаря

3.1. Распознавание языка (language identification)

В настоящий момент система распознает английский, французский, немецкий, русский, эстонский, литовский и латышский языки.

Алгоритм распознавания языка разработан на основе технологии «N-Gram-Based Text Categorization» [6], [7] – статистических данных, полученных из документов, для которых язык и кодировка известны заранее. После подсчета частот N-Gram (последовательность символов не более N), опираясь на гипотезу о том, что приблизительно 300 наиболее часто используемых N-Gram зависят от языка в большой степени, распознаваемый язык будет идентифицирован с известным языком документа при условии минимального расстояния от N-Gram статистики до статистики известного документа.

Модуль распознавания языка необходим не только системе АО текста для ее внутренней организации, но и пользователю системы. При активной функции «Detect source language automatically» пользователю не приходится каждый раз выбирать направление перевода, задавать системе язык-источник (ЯИ) и язык перевода (ЯП). Работа с текстом таким образом становится еще эффективнее с точки зрения экономии времени.

3.2. Синтаксический анализ текста (parsing of source language sentence)

Информация, полученная в результате процедуры распознавания языка, посылается далее на обработку модулю морфологического анализа (МА) и синтаксического анализа (СА) текста (парсинг), в результате которого предложение будет проанализировано полностью или частично. МА встроен в СА и является его неотъемлемой частью.

```
NP -> attr:AP main:N (mod:PP)
        attr:AP.Case==main:N.Case
        attr:AP.Gender==main:N.Gender
        attr:AP.Number==main:N.Number
        NP.Case=main:N.Case
        NP.Gender=main:N.Gender
        NP.Number=main:N.Number
```

Рисунок 3. Пример правила именной группы

Правило парсера состоит из двух частей: описание синтаксической структуры (правило контекстно-свободной грамматики) и условие использования (грамматика правил). Грамматика правил описывает ограничения на использование конструкции, а также назначает либо игнорирует возможную связь между вершинами дерева. Рисунок 3 демонстрирует пример правила именной группы. Правило описывает структуру NP, состоящую из прилагательного AP, существительного N (вершина дерева) и предлога PP. Двойной знак равенства «==» используется при описании условия, то есть, правило будет исполнено только при условии согласования AP в падеже, роде и числе с N. Одинарный знак равенства «=» означает присвоение признака.

Парсер разработан на основе универсального CYK-алгоритма для контекстно-свободной грамматики. Данный алгоритм осуществляет восходящий СА и позволяет частичный разбор предложения ЯИ (Cocke-Younger-Kasami) [8], [9], [10]. Оригинальный CYK-алгоритм поддерживает контекстно-свободную грамматику Хомского. Наша разработка отличается от оригинальной версии дополнением признаков, которые используются в правилах на ограничение, присвоение либо передачу признака другой вершине дерева.

На выходе следующий модуль получает синтаксическое дерево (рисунок 4) или части дерева в случае неуспешного разбора целого предложения. Вершины дерева связаны между собой синтаксическими отношениями, набор которых варьируется от языка к языку. Для английского, французского, немецкого и русского языков технологии парсинга были лицензированы у различных разработчиков, поэтому выходные данные систем отличаются друг от друга в зависимости от производителя и языка и не совпадают. Нами был создан дополнительный компонент, который приводит эти данные к единому формату, необходимому для дальнейшей обработки.

```
ministri nolēma piešķirt līdzekļus vētras seku novēršanai
nolēma        Base form:nolemt Morphology:vs0000300i000000000000000010
    ministri:subj              Base form:ministrs
                               Morphology:nOmpn030000000n000000000000010
    piešķirt:obj               Base form:piešķirt
                               Morphology:v000000000n000000000000000010
        līdzekļus:obj          Base form:līdzeklis
                               Morphology:nOmpa030000000n000000000000010
        novēršanai:dat         Base form:novēršana
                               Morphology:nOfsd030000000n000000000000010
            seku:mod           Base form:sekas
                               Morphology:nOfpg030000000n000000000000010
                vētras:mod
              Base form:vētra Morphology:nOfsg030000000n000000000000010
```

Рисунок 4. Предложение латышского языка на выходе парсера

Для литовского и латышского языков модули парсинга были разработаны в рамках проекта. Для эстонского языка на сегодняшний день есть небольшая демонстрационная версия грамматики и набор синтаксических правил, который используется эстонской компанией Filosoft.

Для РЯ были лицензированы технологии МА Андрея Коваленко [11] и СА DictaScope [12].

3.3. Обработка устойчивых словосочетаний (idiom detection)

С точки зрения МП устойчивые словосочетания должны обрабатываться как единое целое. В нашей системе устойчивые словосочетания хранятся в отдельном словаре и анализируются как одна вершина дерева. На сегодняшний день при переводе устойчивых конструкций не учитывается информация о структуре всего дерева

в целом, однако словосочетание интегрируется в синтаксическое дерево и учитывается при дальнейшей обработке. Подобный подход используется в англо↔латышском переводе. Начаты работы по созданию модуля для латышско↔русского перевода. Исследуется возможность обработки компонентов внутри структуры словосочетания. Можно сказать, что модуль устойчивых словосочетаний находится в начальной стадии разработки, многое еще должно быть сделано.

3.4. Синтаксический трансфер (syntactic transfer)

На стадии трансфера дерево входного предложения ЯИ трансформируется в дерево выходного предложения ЯП с помощью трансформационных правил ТП. ТП могут быть следующего вида:

1. установление связи между вершинами дерева выходного предложения
2. изменение порядка слов
3. удаление или скрытие вершины дерева
4. добавление новой вершины
5. перенос или назначение синтаксической, морфологической или лексической информации
6. изменение типа синтаксического отношения между вершинами дерева

Обычно ТП применяются к двум (реже трем) синтаксически связанным вершинам.

ТП разработаны для всех направлений перевода. В настоящем докладе мы остановимся на латышско↔русском трансфере, для реализации которого детально исследовались функциональные свойства обоих языков.

Одной из основных проблем латышско↔русского перевода является гипертрофия родительного падежа (ģenitīvs) в ЛЯ. Для решения данной проблемы были детально изучены атрибутивные конструкции ЛЯ с целью классификации. Согласованные определения не представляют больших трудностей для системы, так как с помощью правил согласования, которые будут описаны ниже, зависимому прилагательному, местоимению, числительному или причастию присваиваются все признаки главного существительного, например:

(1) balta kleita – белое платье
(2) liels galds – большой стол
(3) mans brālis – мой брат
(4) tie koki – те деревья
(5) sava māja – свой дом
(6) pirmais zvans – первый звонок
(7) ziedošs koks – цветущее дерево

На этапе трансфера в данном случае требуется назначить тип связи между вершинами дерева (правило типа 1). На рисунке 5 дан пример ТП, в котором функция MakeLink назначает тип синтаксического отношения attr между прилагательным и существительным.

Сложнее дело обстоит с несогласованными определениями типа NgenN (существительное в родительном падеже + существительное), например:

(1) studentu grupa – группа студентов (количественное значение)
(2) direktora kabinets – кабинет директор / директорский кабинет, lapsas aste – лисий хвост / хвост лисы (притяжательное значение)
(3) kustības ātrums – скорость движения (значение носителя признака)
(4) koka galotne – верхушка дерева (значение содержания часть-целое)
(5) tēva cieņa – уважение отца (объектное значение), drauga zvans – звонок друга (субъектное значение)
(6) stikla glāze, ābolu sula, koka galds, lapsas aste – стеклянный стакан, яблочный сок, деревянный стол (значение вещества или материала),
(7) litrs piena, karote cukura – литр молока, ложка сахара/у (значение партитива)

```
TransferRule(A-attr->N) //skaista grāmata
{
    MakeLink(Child-attr->Parent);
}
```

Рисунок 5. Правило типа 1 назначает связь между компонентами именной группы

Во всех приведенных выше примерах ЛЯ генитив находится в препозиции к определяемому существительному, что нельзя сказать о РЯ, в котором генитив находится в постпозиции к главному существительному (пример 1-5). С помощью ТП типа 2 переводятся генитивные конструкции (рисунок 6).

```
TransferRule(N<-mod-N) //lasītāju skaits
{
    move_to_right(Child,Parent);
    MakeLink(Child-mod->Parent);
}
```

Рисунок 6. Правило типа 2, ставящее зависимое слово в постпозицию к определяемому

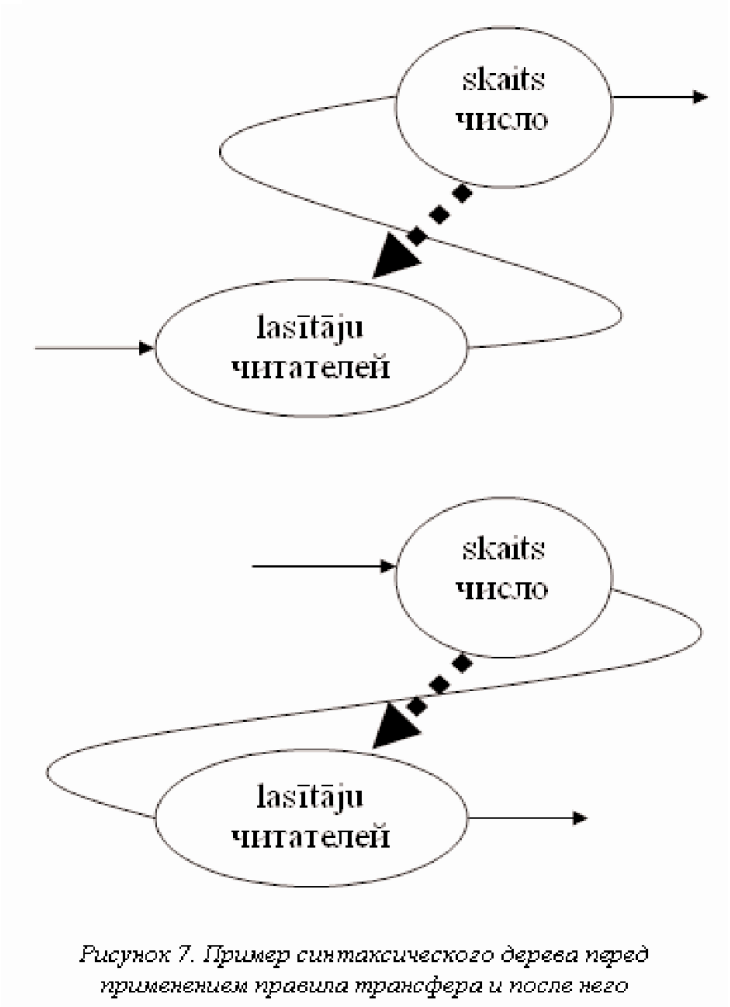

Рисунок 7. Пример синтаксического дерева перед применением правила трансфера и после него

При выполнении данного правила слово «lasītāju» будет перемещено в положение постпозиции по отношению к главному слову «skaits». За данную процедуру отвечает функция *move_to_right*. На рисунке 7 жирной пунктирной линией отражено направление связи между двумя вершинами дерева. Тонкой сплошной линией отражен порядок слов во входной конструкции ЛЯ (верхнее изображение) до исполнения ТП и порядок слов в выходной конструкции РЯ (нижнее изображение) после исполнения ТП.

Ситуация осложняется тем, что не всегда подобные конструкции будут переводиться одним способом. Примеры 1, 3-5 переводятся конструкцией NNgen. Примеры 2 и 6 демонстрируют другой случай, когда в ЛЯ существительное в генитиве переводится относительным или притяжательным прилагательным в РЯ (во втором примере возможны оба варианта). Русским относительным и притяжательным прилагательным соответствует форма генитива латышского существительного. В обоих случаях у нас есть выбор:

(1) lapsas aste – лисий хвост / хвост лисы

(2) ābolu sula – яблочный сок / сок из яблок

Как видно из примеров, мы можем перевести с помощью конструкций AN, NNgen либо NPN. Однако, чтобы выбрать перевод той или иной конструкции, мы должны задать способ формализации значений генитива в ЛЯ.

Другой пример, в котором существительное ЛЯ в генитивной конструкции переводится по-разному на РЯ:

(*1) koka galds / koka galotne – деревянный стол / верхушка дерева*

(2) stikla glāze / stikla ražošana – стеклянный стакан / производство стекла

Выбор наиболее подходящего перевода подобных генитивных конструкций с помощью ТП на сегодняшний день не осуществлен. В ближайшем будущем мы планируем использовать статистические методы для решения данной задачи.

Типичный пример трансформационного правила типа 3, когда необходимо удалить или скрыть вершину дерева, – перевод глагола в форме сослагательного наклонения с РЯ на ЛЯ:

(1) хотел бы – gribētu

Рисунок 8 демонстрирует правило подобного типа, когда функция *MakeLink* назначает тип синтаксического отношения *hidden* между глаголом и частицей *бы*, что позволяет ее скрыть в синтаксической структуре ЛЯ.

Данное правило может также служить примером правила типа 6, изменения типа синтаксического отношения между вершинами дерева: во входной синтаксической структуре две вершины соединены синтаксическим отношением *phr*, в выходной структуре – *hidden*, а также примером правила типа 5, назначение морфологической информации.

Одна из трудностей связана с категорией глагола. Так, в ЛЯ 3-е лицо единственного и множественного числа не различается по своей форме:

(1) viņš, viņa, viņi, viņas lasa – он/она читает, они читают

(2) viņš, viņa, viņi, viņas strādāja – он/она работал/работала, они работали

(3) viņš, viņa, viņi, viņas zīmēs – он/она будет рисовать, они будут рисовать

```
TransferRule(V<-phr-PARTICLE) //gribētu
{
    Child.SourceBaseform == "бы";
    Parent.Time == past;
    Parent.Time = present;
    Parent.Gender = nothing;
    Parent.Mode = subjunctive;
    Parent.Reflexive = nothing;
    Parent.Person = nothing;
    Parent.Number = nothing;
    MakeLink(Child-hidden->Parent);
}
```

Рисунок 8. Пример трансформационного правила типа 3

В примерах 1, 2 мы видим, что в ЛЯ формы глагола 3-его лица единственного и множественного числа не различаются. Пример 2 демонстрирует тот факт, что в ЛЯ категория рода не свойственна глаголу. При переводе на РЯ мы должны перенести соответствующие значения числа и рода с местоимения в данном случае на глагол, чтобы далее был осуществлен корректный перевод на РЯ. Пример подобного ТП на рисунке 9.

```
TransferRule(PRON-subj->V)
{
    Parent.Transport.Person = Child.Person;
    Parent.Transport.Gender = Child.Gender;
    Parent.Transport.PronounType = Child.PronounType;
    Parent.Transport.Number = Child.Number;
    Parent.Transport.Case = Child.Case;
    MakeLink(Child-subj->Parent);
}
```

Рисунок 9. Правило типа 5: перенос значений местоимения на глагол

ТП типа 4 разработаны с целью добавления новой вершины в структуру синтаксического дерева. Примером такого ТП может служить перевод конструкции типа V-obj->N:

(1) gatavoties (kam? Datīvs) eksāmeniem – готовиться к (чему? Творительный падеж) экзаменам

ТП добавления предлога с помощью функции *insert_node* отображено на рисунке 10. На сегодняшний день проблема различий в управлении глагола в ЛЯ и РЯ решается с помощью ТП для отдельных случаев. Параллельно изучается возможность и разрабатывается алгоритм решения этой задачи централизованно.

```
TransferRule(V<-dat-N) //gatavoties
{
    Parent.SourceBaseform == "gatavoties";
    insert_right(NewNode,Parent,pcomp);
    NewNode.TargetSpelling = "к";
    NewNode.POS = PREP;
    MakeLink(Child-dat->Parent);
}
```

Рисунок 10. Правило типа 4: добавление новой вершины

Также классическим примером ТП типа 3 и 5 является перевод глаголов с отрицательной частицей с РЯ на ЛЯ, так как в ЛЯ она пишется слитно с глаголом и является отрицательной приставкой (рисунок 11).

3.5. Перевод (lexical transfer)

Данный модуль программы осуществляет собственно перевод выходной структуры трансфера, находя переводной эквивалент (ПЭ) в двуязычном словаре и основываясь на характеристику части речи (ЧР). Например, при переводе конструкции *es zīmēju ovālu* существительное *овал* будет выбрано, а прилагательное *овальный* отклонено.

```
TransferRule(PARTICLE-neg->V) //nebūs
{
    Parent.SourceBaseform == "быть";
    Child.SourceBaseform == "не";
    Parent.Transport.NegativePrefix = negative;
    MakeLink(Child-hidden->Parent);
}
```

Рисунок 11. Правило типа 3 и 5: скрытие вершины дерева и присвоение значения

Если в словаре не находится ПЭ соответствующей ЧР, тогда делается попытка найти его из альтернативных ЧР. Такое встречается из-за разночтения в различных системах МА, например, морфология ЛЯ анализирует слово *mans* как местоимение, а морфология РЯ – как прилагательное. В результате вместо местоимения трансфер найдет соответствующее прилагательное.

С помощью лексического трансфера в ходе определенных преобразований также переводятся такие формы слов, которых обычно нет в словаре, например, причастие сначала трансформируется в инфинитив, потом переводится, а далее из инфинитива ЯП синтезируется необходимая форма причастия:

(1) smaidošs – smaidīt – улыбаться – улыбающийся

3.6. Разрешение многозначности (disambiguation)

В системе используется статистический метод разрешения многозначности. В действующей системе представлены модули разрешения многозначности для латышского и литовского языков.

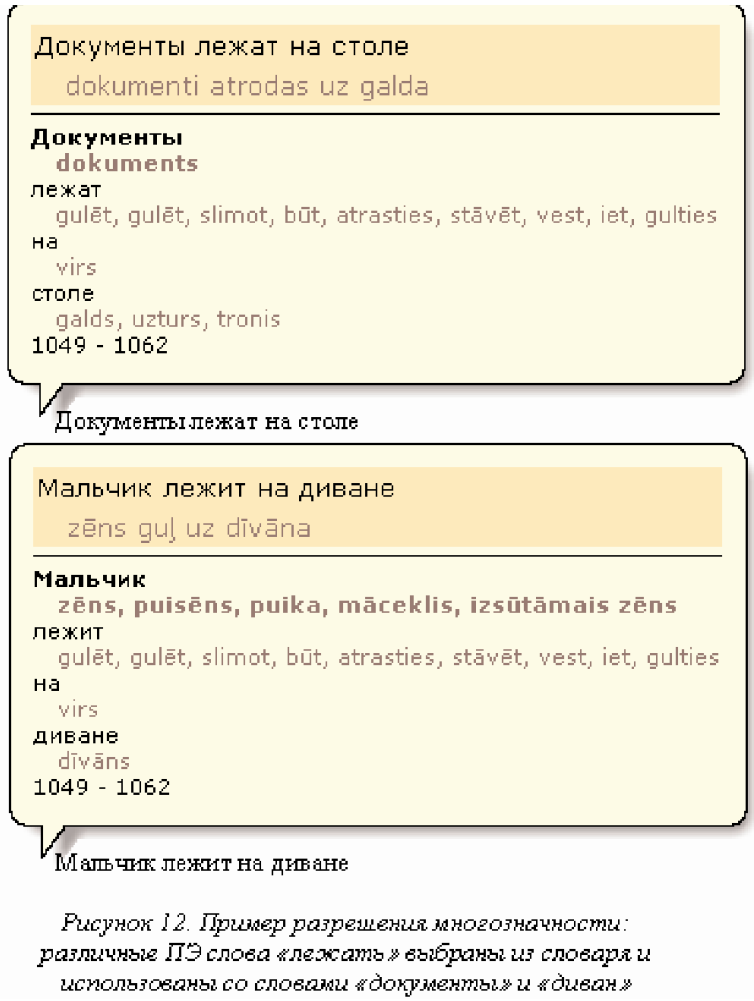

Рисунок 12. Пример разрешения многозначности: различные ПЭ слова «лежать» выбраны из словаря и использованы со словами «документы» и «диван»

В направлении перевода с РЯ на ЛЯ мы руководствуемся статистическими данными о вероятности синтаксических пар – связи двух слов во фразе или предложении определенным синтаксическим отношением. Этот метод нам представляется более эффективными по сравнению с методом биграмм – вероятностью нахождения двух слов рядом в предложении. Мы используем синтаксические отношения *subject(NV), object(VN), attribute(AN) и attribute(NN)*. Из корпуса текстов ЛЯ с помощью парсера были выбраны синтаксически связанные пары слов. Частота каждой уникальной пары была вычислена, после чего была вычислена также вероятность появления данной синтаксической пары. Полученные результаты были названы *синтаксическая модель языка (СМЯ)* и использованы в модуле разрешения многозначности.

В дереве ЯП мы имеем один или несколько ПЭ, соотнесенных с одной вершиной дерева ЯИ. Для каждой пары вершин, связанных в дереве ЯП, мы узнаем вероятность ее появления из полученной нами СМЯ. Таким образом, мы имеем возможность разрешения многозначности, выбирая те ПЭ, которые обеспечивают наивысшую вероятность.

Пример разрешения многозначности, основанного на использовании данного метода демонстрирует рисунок 12.

3.7. Согласование (agreement)

В результате работы модуля разрешения многозначности каждая вершина дерева ЯИ представлена одним ПЭ, которому назначены соответствующие морфологические характеристики ЯИ, часто не совпадающие с характеристиками ЯП. За координирование последних отвечает модуль согласования.

Типичным примером применения правил согласования (ПС) в ЛЯ↔РЯ является перенос значения рода существительного в следующих случаях:

(1) существительные среднего рода РЯ: sviests (м.р.) – масло (ср.р.)

(2) значение рода не совпадает у ПЭ: diena (ж.р.)↔день (м.р.), sols (м.р.)↔ парта (ж.р.)

Пример ПС демонстрирует рисунок 13.

```
Rule(N) // māja/дом, zirgs/лошадь utt.
{
    Target.Gender = Analyze.Gender;
    Target.Person = nothing;
    Target.Deminutive = nothing;
}
```

Рисунок 13. Пример правила согласования

В результате выполнения данного правила вершине дерева будет присвоено значение рода ПЭ.

```
Rule(V<-comp-N) //[viņš] ir skolnieks
{
    Parent.SourceBaseform == "būt";
    Parent.Time == present;
    Parent.Mode == indicative;
    Child.Case == nominative;
    Parent.Transport.PronounType == person;
    Parent.TargetSpelling = "";
}
```

Рисунок 14. Пример правила согласования

Проблема согласования между именной группой и глагольной также решается на данном этапе. С помощью ТП соответствующие значения были перенесены на глагол (рисунок 9). Теперь на этапе согласования данные значения будут использованы в синтезе именной группы (рисунок 14).

3.8. Синтез (output generation)

Последний этап работы словаря – оформление результатов перевода. Модуль синтеза или генерации возвращает перевод пользователю. Рисунки 15, 16 демонстрируют ЛЯ↔РЯ и РЯ↔ЛЯ перевод.

4. Заключение

Представленная в докладе система является пилотной версией, однако действующей. Многое еще предстоит разработать и добавить в будущем. Это касается не только алгоритмов (анализ и синтез текста), но и содержания (словарная информация). На сегодняшний день активно ведутся работы по добавлению семантического фильтра, а также над использованием статистических методов в помощь там, где не представляется возможным решение задачи с помощью трансфера.

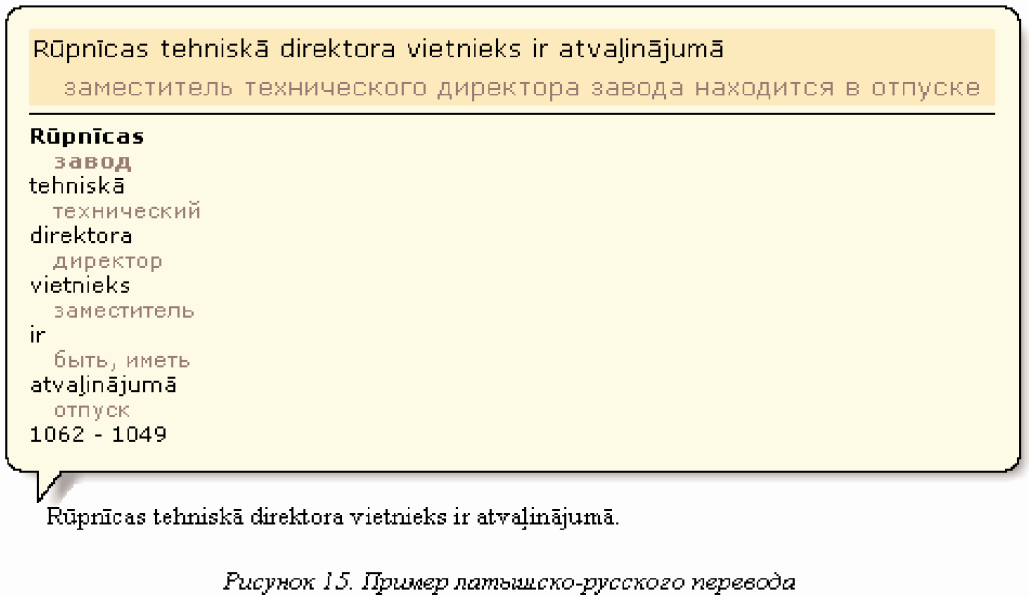

Рисунок 15. Пример латышско-русского перевода

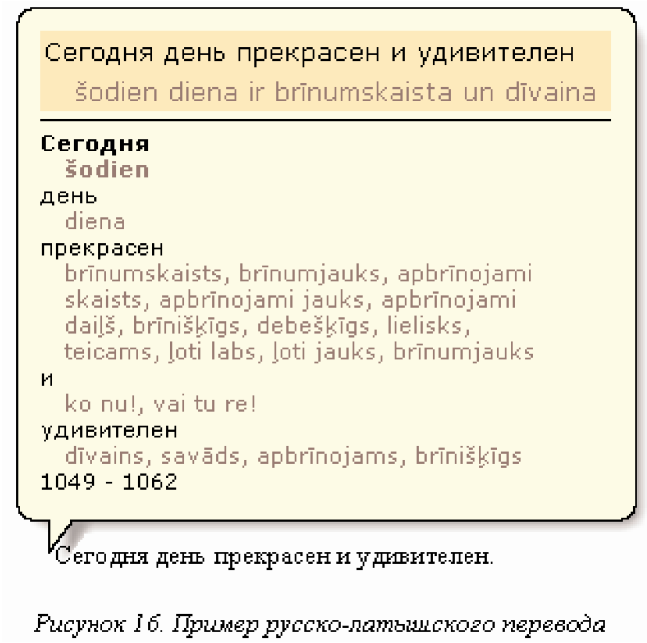

Рисунок 1 б. Пример русско-латышского перевода

Список литературы

1. Vasiljevs A., Skadina I., Deksne D., Skadins R. Human Language Technologies for Baltic Languages – Developments and Perspectives // Proceedings of Workshop on International Proofing Tools and Language Technologies, Patras, Greece, 2004.
2. Deksne D., Skadiņa I., Skadiņš R., Vasiļjevs A. Foreign language reading tool – first step towards English-Latvian commercial machine translation system // Proceeding of the Second Baltic Conference on Human Language Technologies, Tallinn, 2005.
3. Vasiļjevs A., Ķikāne J., Skadiņš R. Development of HLT for Baltic languages in widely used applications // Proceedings of First Baltic Conference „Human Language Technologies – the Baltic Perspective", Riga, 2004.
4. Prószéky G., Balázs K. Development of a Context-Sensitive Dictionary // Proceedings of the 10th International Congress of the European Association for Lexicography (EURALEX), Copenhagen, Denmark, 2002.
5. Feldweg H., Breidt E. COMPASS - An Intelligent Dictionary System for Reading Text in a Foreign Language // Papers in Computational Lexicography (COMPLEX 96), Linguistics Institute, HAS, Budapest, 1996.
6. Grefenstette G. Comparing two Language Identification Schemes // JADT 1995, 3rd International conference on Statistical Analysis of Textual Data, Rome, 1995.
7. Cavnar W.B., Trenkle J.M. N-Gram-Based Text Categorization // Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, 1994.
8. Cocke J., Schwartz Jacob T. Programming languages and their compilers // Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.
9. Kasami T. An efficient recognition and syntax-analysis algorithm for context-free languages // Scientific report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA, 1965.
10. Younger Daniel H. Recognition and parsing of context-free languages in time n3 // Information and Control 10(2), 1967.
11. электронный ресурс: http://www.linguist.nm.ru/ling/rus/help.htm
12. электронный ресурс: http://www.dictum.ru/?main=products&sub=dictascope