Процедуры упрощения предложений на естественном языке на основе синтаксической разметки CoNLL-U для задач TextMining

Т. С. Гаршин, email: timur.garshin@gmail.com
А. Ю. Иванков, email: ivankov@cs.vsu.ru
Е.В. Муравьёва, email: ketmur18@gmail.com1

Воронежский государственный университет

Аннотация. *В статье приводится подход к задаче извлечения фактов из сложного предложения русского языка, основанный на анализе синтаксической разметки предложения. Представлены алгоритмы упрощения предложения путем удаления из него деепричастного, причастного оборотов и прямой речи на основе разбора синтаксической разметки CoNLL-U.*

Ключевые слова: *Лингвистический анализ, синтаксический парсинг, CoNLL-U, TextMining .*

Введение

В технологиях TextMining одной из актуальных задач является извлечение фактов и данных из массивов неструктурированных текстовых документов. Решения этой задачи в настоящее время строятся на основе двух подходов:

использования обученных нейросетей для поиска именованных сущностей в тексте без учета синтаксиста

применение алгоритмов разбора предложений (как нейросетевых, так и основанных на правилах), учитывающих морфологию и синтаксис.

Каждый из них имеет свои достоинства и недостатки. Так, нейросетевой подход, реализованный в достаточно большом числе систем, дает хорошие результаты по выявлению отдельных слов и словосочетаний, но есть сложности при адаптации на новую предметную область текстов. Подходы, основанные на лингвистических правилах, позволяют провести более детальный разбор предложений, но обладают большей сложностью реализации.

1. Пути устранения ошибок при выделении фактов из предложения

Рассмотрим наиболее частую причину ошибок при разборе словосочетаний и групп слов, определяющих какой-либо искомый факт. Например, в предложении – Компания Леонардо Ди Каприо Appian Way заключили эксклюзивный контракт с Apple. Нейросеть Bert (ner_ontonotes_bert_mult) выделяет “Леонардо Ди Каприо” как наименование организации, это происходит из-за соседства со словом “компания”, которое идентифицируется нейросетью как организация. Если провести синтаксический разбор этого же предложения, то мы сможем проанализировать маркированные синтаксические связи, позволяющие выделить группу слов, относящихся к названию компании. Синтаксический разбор этого предложения, реализованный с помощью парсера deepavlov, представлен на рисунке 1. Формат представления содержит табличный вид и вид дерева разбора с разметкой связей в формате Conll-u. Это широко применяемый формат представления синтаксических зависимостей для решения задач в области компьютерной лингвистики. Он основан на соединении формального представления универсальных зависимостей Стэнфордского синтаксического анализатора (Universal Dependencies), разработанного для английского языка и универсальных тегов частей речи Google, адаптированных для различных языков.

Формат разбора предложения в Conll-u имеет следующую структуру [1] – содержит одну или несколько строк слов, формирующих предложение, а строки слов содержат следующие поля:

1. ID: индекс слова, целое число, начиная с 1 для каждого нового предложения; может быть диапазоном токенов с несколькими словами.
2. FORM: словоформа или знак препинания.
3. LEMMA: Лемма или основа словоформы.
4. UPOSTAG: универсальный тег части речи.
5. XPOSTAG: тег части речи для конкретного языка.
6. FEATS: список морфологических характеристик.
7. HEAD: заголовок текущего токена, который является либо значением ID, либо нулем (0).
8. DEPREL: Universal Stanford dependency relation к (root iff HEAD = 0) или определенному зависящему от языка подтипу.
9. DEPS: Список вторичных зависимостей.
10. MISC: любая другая аннотация.

Ввод (редактируемый):

```
1       Компания        _       _       _       _       7       nsubj   _       _
2       Леонардо        _       _       _       _       1       nmod    _       _
3       Ди      _       _       _       _       2       flat:name       _       _
4       Каприо  _       _       _       _       2       flat:name       _       _
5       Appian  _       _       _       _       1       flat:foreign    _       _
6       Way     _       _       _       _       1       flat:foreign    _       _
7       заключили       _       _       _       _       0       root    _       _
8       эксклюзивный    _       _       _       _       9       amod    _       _
9       контракт        _       _       _       _       7       obj     _       _
10      с       _       _       _       _       11      case    _       _
11      Apple   _       _       _       _       9       nmod    _       _
12      .       _       _       _       _       7       punct   _       _
```

Рис. 1. Пример синтаксического разбора предложения парсером deepavlov с разметкой Conll-u.

2. Подход к выделению фактов из предложений русского языка

Для повышения эффективности извлечения фактов из русскоязычных текстов, предлагается следующий гибридный подход, основанный на интеграции существующих открытых технологий обработки текста и разработанных нами процедур (Рис.2).

Рис. 2. Общая архитектура решения гибридного подхода извлечения фактов из текста

В проекте используется DeepPavlov - фреймворк для анализа текста и создания диалоговых систем с открытым исходным кодом для проведения синтаксического разбора текущего предложения и получение его описания в формате Conll-u. Входное предложение русского языка с помощью DeepPavlov проводится через синтаксический парсинг и для него строится дерево зависимостей с разметкой Conll-u. Затем, разработанные процедуры упрощения

предложений, реализованные на языке Python, реализуют парсинг размеченных предложений, с целью отсечения отдельных ситаксических связей. В результате формируется набор упрощенных предложений, к которым применяется набор разработанных КС-грамматик Томита-Парсера для извлечения фактов. Данная архитектура решения позволяет упростить процесс и сократить количество ошибок при извлечении фактов из предложений.

3. Описание процесса обработки текста

Задача упрощения предложений связана с реализацией алгоритмов обработки сложных предложений с учетом синтаксической разметки Conll-u. Рассмотрим реализованные варианты обработки.

Обработка деепричастного\причастного оборота:

1. Предложение преодобработанное deepavlov подается на вход и разбирается как список и древовидная структура.
2. Разобранное деревом предложение поступает на обработку в рекурсивную функцию, которая создает два списка: первый – все дочерние узлы дерева, второй – все узлы являющиеся дочерними от искомой связи (в случае деепричастного оборота - это связь advcl, в случае причастного - acl).
3. Из общего списка всех узлов дерева удаляются узлы, промаркированные как дочерние от искомой связи.
4. Оставшиеся элементы списка сортируются по возрастанию значения ID, и таким образом, формируется генерация выделяемого предложение.

Обработка предложения, содержащего прямую речь :

1. Предложение преодобработанное deepavlov подается на вход и разбирается как список и древовидная структура.
2. Разобранное предложение (синтаксическое дерево) поступает на обработку в рекурсивную функцию, которая создает два списка: первый – все дочерние узлы дерева, второй – все узлы являющиеся дочерними от искомой связи (в случае прямой речи это - parataxis).
3. Формируется новая переменная, в которую записываются все элементы списка: в первый - не вошедшие, во второй список с подписью («автор прямой речи») и отдельно с подписью («прямая речь») выводится сам второй список.
4. Оставшиеся элементы списка сортируются по возрастанию значения ID, и таким образом, по ним генерируется выделяемого предложение.

Таблица 1

Обработка прямой речи

Исходное\итоговое предложение	Код разбора предложения

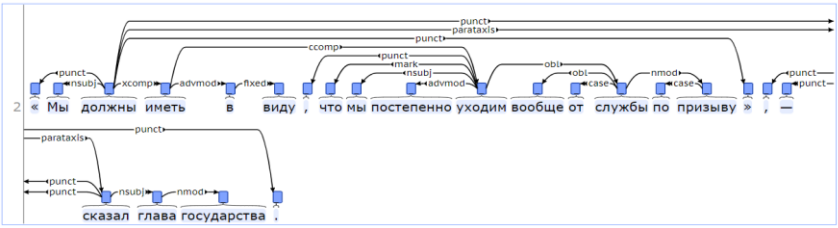
Исходное предложение : "Мы должны иметь в виду , что мы постепенно уходим от службы по призыву", сказал глава государства.
text ="""
1 « _ _ _ _ 3 punct _ _
2 Мы _ _ _ _ 3 nsubj _ _
3 должны _ _ _ _ 0 root _ _
4 иметь _ _ _ _ 3 xcomp _ _
5 в _ _ _ _ 4 advmod _ _
6 виду _ _ _ _ 5 fixed _ _
7 , _ _ _ _ 11 punct _ _
8 что _ _ _ _ 11 mark _ _
9 мы _ _ _ _ 11 nsubj _ _
10 постепенно _ _ _ _ 11 advmod _ _
11 уходим _ _ _ _ 4 ccomp _ _
12 вообще _ _ _ _ 14 obl _ _
13 от _ _ _ _ 14 case _ _
14 службы _ _ _ _ 11 obl _ _
15 по _ _ _ _ 16 case _ _
16 призыву _ _ _ _ 14 nmod _ _
17 » _ _ _ _ 3 punct _ _
18 , _ _ _ _ 20 punct _ _
19 — _ _ _ _ 20 punct _ _
20 сказал _ _ _ _ 3 parataxis _ _
21 глава _ _ _ _ 20 nsubj _ _
22 государства _ _ _ _ 21 nmod _ _
23 . _ _ _ _ 24 punct _ _
 	
"""
Результат обработки:
Прямая речь- TokenList<« Мы должны иметь в виду что мы постепенно уходим вообще от службы по призыву » .> Автор прямой речи- TokenList< — сказал глава государства>
	# обычный разбор предложения, не древовидный, понадобится в конце
	parse_norm = parse(text)[0]
	pars_not_norm = parse(text) [0]
	save_str =[]
	# древовидный разбор
	root = parse_norm.to_tree()
	# список связей, за которыми необходимо следить
	deprel_needed = ['parataxis']
	# словарь слов, которые надо удалить (деепричастный оборот), ключ - id, значение - само слово
	words_to_remove = {}
	words_to_save={}
	# рекурсивная фия обхода всех веток, учитывая связь
	def parse_syntactic_tree(children, mark_all_children=False):
	
	  for i in children:
	  if i.token['deprel'] in deprel_needed or mark_all_children:
	  words_to_remove[i.token['id']] = i.token['form']
	
	  parse_syntactic_tree(i.children, mark_all_children=True)
	  else:
	  parse_syntactic_tree(i.children)
	
	def remove_words():
	
	  # удаляем все ненужные слова, которые нашли
	  for i in words_to_remove:

```
    parse_norm.remove([x for x in parse
_norm if x['id'] == i][0])

  parse_norm_to_str= str(parse_norm)
  valids = re.sub(r"[',']*", ",  parse_nor
m_to_str)
  print('Прямая речь-',valids)
    for j in parse_norm:
    pars_not_norm.remove(j)

  pars_not_norm_to_str= str(pars_not_n
orm)
  valids = re.sub(r"[',']*", ",  pars_not_n
orm_to_str)
  print('Автор прямой речи-',valids)

def create_sentence():
# сборка фразы
  result_str = ''
  for s in parse_norm:
    if s['form'] in '.,?!:;-' or not result_str:
      result_str += s['form']
    else:
      result_str +=  f" {s['form']}"
```



Ввод (редактируемый):

```
17      »       _       _       _       _       3       punct   _       _
18      ,       _       _       _       _       20      punct   _       _
19      —       _       _       _       _       20      punct   _       _
20      сказал  _       _       _       _       3       parataxis       _
.
21      глава   _       _       _       _       20      nsubj   _       _
22      государства     _       _       _       _       21      nmod    _
.
23      .       _       _       _       _       3       punct   _       _
```

Рис. 3. Визуализация разобранного предложения с прямой речью библиотекой conllu.

Таблица 2

Причастный оборот

Исходное\итоговое предложение	Код разбора предложения

#исходный текст -
Русский библиографический институт, из года в год определяющий "Людей года", обнародовал список "Памятных людей десятилетия".
text= """
1 Русский _ _ _ _ 3 amod _ _
2 биографический _ _ _ _ 3 amod _ _
3 институт _ _ _ _ 15 nsubj _ _
4 , _ _ _ _ 9 punct _ _
5 из _ _ _ _ 6 case _ _
6 года _ _ _ _ 9 obl _ _
7 в _ _ _ _ 8 case _ _
8 год _ _ _ _ 6 nmod _ _
9 определяющий _ _ _ _ 3 acl _ _
10 `` _ _ _ _ 11 punct _ _
11 Людей _ _ _ _ 9 obj _ _
12 года _ _ _ _ 11 nmod _ _
13 " _ _ _ _ 11 punct _ _
	parse_norm = parse(text)[0]
	root = parse_norm.to_tree()
	deprel_needed = ['acl']
	# словарь слов, которые надо удалить причастный оборот), ключ - id, значение - само слово
	words_to_remove = {}
	# рекурсивная ф-ия обхода всех веток, учитывая связь
	def parse_syntactic_tree(children, mark_all_children=False):
	  for i in children:
	  if i.token['deprel'] in deprel_needed or mark_all_children:
	  words_to_remove[i.token['id']] = i.token['form']
	  parse_syntactic_tree(i.children, mark_all_children=True)

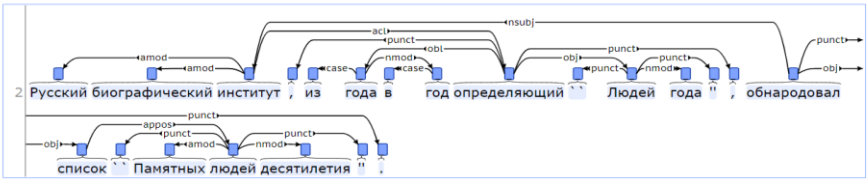
```
14  , _ _ _ _ 9 punct _ _
15  обнародовал _ _ _ _ 0 root  _ _
16  список  _ _ _ _ 15  obj _ _
17  ``  _ _ _ _ 19  punct _ _
18  Памятных  _ _ _ _ 19  amod  _ _
19  людей _ _ _ _ 16  appos _ _
20  десятилетия _ _ _ _ 19  nmod  _ _
21  "  _ _ _ _ 19  punct _ _
22  . _ _ _ _ 15  punct _ _

"""
результат разбора:
Русский биографический институт
обнародовал
список `` Памятных людей десятилетия
".
```

```
    else:
      parse_syntactic_tree(i.children)

def remove_words():
    for i in words_to_remove:

    parse_norm.remove([x for x in parse
_norm if x['id'] == i][0])
   # parse_norm.remove([x for x in pars
e_norm if x['form'] == ','])
def create_sentence():
  result_str = ''
  for s in parse_norm:
    if s['form'] in '.,?!:;-' or not result_str:
      result_str += s['form']
    else:
      result_str +=  f" {s['form']}"
  return result_str
```



Ввод (редактируемый):

```
16      список  _       _       _       _       15      obj     _       _
17      ``      _       _       _       _       19      punct   _       _
18      Памятных        _       _       _       _       19      amod    _       _
19      людей   _       _       _       _       16      appos   _       _
20      десятилетия     _       _       _       _       19      nmod    _       _
21      ''      _       _       _       _       19      punct   _       _
22      .       _       _       _       _       15      punct   _       _
```

Рис. 4. Визуализация разобранного предложения с причастным оборотом библотекой conllu.

Аналогичным образом проводится разбор деепричастного оборота, значимым отличием будет другая искомая связь(advcl).

Заключение

В статье описан подход к реализации задачи извлечения фактов из текста, основанный на интеграции существующих открытых технологий

обработки текста и разработанных процедур. Представленый метод упрощения предложений может быть использован для анализа других синтаксических конструкций русского языка.

Задачи синтаксического парсинга связаны с проблемой семантической интеграции добытых фактов, а также проведением процедур смысловой аналитики. В настоящее время примеров успешной реализации таких систем крайне мало, особенно для русского языка. Практическим применением подобных решений могут быть информационные системы сбора и обработки новостей и сообщений в сети Интернет, систематизации корпоративной текстовой информации, проведение конкурентной разведки и др.

Список литературы

1. Описание проекта: Синтаксический Анализатор CoNLL-U. [Электронный ресурс]. – Режим доступа : https://pypi.org/project/conllu/

2. Python3 для начинающих Списки (list). Функции и методы списков [Электронный ресурс]. – Режим доступа : https://pythonworld.ru/tipy-dannyx-v-python/spiski-list-funkcii-i-metody-spiskov.html

3. Открытый фреймворк для анализа текста и создания диалоговых систем. [Электронный ресурс]. – Режим доступа : https://deeppavlov.ai/

4.