УДК 004.82 https://doi.org/10.15407/pp2020.01.025

К.А. Кудим, Г.Ю. Проскудина

ОБ ОДНОМ МЕТОДЕ ИЗВЛЕЧЕНИЯ ДАННЫХ ИЗ СЛАБОСТРУКТУРИРОВАННЫХ ДОКУМЕНТОВ

В работе разработан, подробно описан и практически опробован лингвистический метод решения задачи извлечения данных на примере извлечения данных о персоналиях из слабоструктурированных документов, представленных в общедоступном каталоге авторефератов диссертаций Национальной библиотеки Украины им. В.И. Вернадского. Описана вся последовательность шагов: выбор коллекции документов; подготовка документов; написание правил грамматики для извлечения данных из текста; написание правил проверки морфологии; создание интерпретаций или привязок правил к данным; анализ результатов разбора. Лингвистический метод извлечения выявил ряд преимуществ по сравнению с описанным ранее методом извлечения данных с помощью регулярных выражений.

Ключевые слова: слабоструктурированные документы, извлечение информации, лингвистический анализатор, синтаксический анализатор, морфологический анализ, контекстно-свободная грамматика.

Введение

Ранее нами был описан метод извлечения данных из слабоструктурированных текстов с помощью регулярных выражений [1]. В настоящей работе метод усовершенствован заменой регулярных выражений на полноценный синтаксический анализатор. А также существенно улучшен метод предварительной подготовки документов.

В работе разработан, подробно описан и практически апробирован лингвистический метод решения задачи извлечения данных на примере извлечения данных о персоналиях из авторефератов диссертаций. Описана вся последовательность шагов: выбор коллекции документов; подготовка документов; написание правил грамматики для извлечения данных из текста; написание правил проверки морфологии; создание интерпретаций или привязок правил к данным; анализ результатов разбора. Лингвистический метод извлечения выявил ряд преимуществ по сравнению с описанным ранее методом извлечения данных с помощью регулярных выражений.

1. Коллекция документов

Для разработки и тестирования метода необходимо прежде всего определиться с коллекцией документов. В качестве типового документа было решено остановиться на авторефератe диссертации. Такой документ имеет ряд необходимых признаков слабоструктурированного документа. В частности, в документе содержится большое количество метаданных о нём самом. Это не только имя автора и название, а также организации, научные руководители и оппоненты, тема диссертации, научное звание и специальность.

В качестве тестовой коллекции авторефератов диссертаций для исследования была выбрана коллекция полнотекстовых документов электронного каталога авторефератов диссертаций Национальной библиотеки Украины им. В.И. Вернадского. В корпусе авторефератов диссертаций представлено более чем 63000 документов. Коллекция состоит из всех авторефератов поданных в Украине за период 1998 – 2011 гг. Полные тексты авторефератов содержатся в файлах преимущественно в форматах RTF и DOC. Коллекция доступна через навигацию по каталогу в виде ссылок в метаописании каждого автореферата. Для обработки на локальной машине была создана программа-краулер, которая обходит электронный каталог используя навигационные ссылки, и скачивает файлы содержащие полный текст. В результате вся коллекция доступна для дальнейшей обработки в локальном каталоге.

2. Подготовка документа

Перед непосредственным извлечением данных исходные файлы обрабатывались в два этапа.

© К.А. Кудим, Г.Ю. Проскудина, 2020

На первом шаге подготовки документа из файлов различных форматов извлекается текст. С помощью готовых утилит операционной системы документы, такие как DOC и RTF, легко преобразуются в простой неструктурированный текстовый формат TXT. Такой же подход можно обнаружить в зрелых программных продуктах для систем электронных библиотек EPrints и DSpace, которые используют подобное преобразование с помощью внешних программ для полнотекстовой индексации документов сложных для анализа форматов. В настоящей работе использовалась утилита textutil из операционной системы macOS. Пример консольной команды для преобразования файла показан на рис. 1.

```
textutil -convert txt file.doc
```

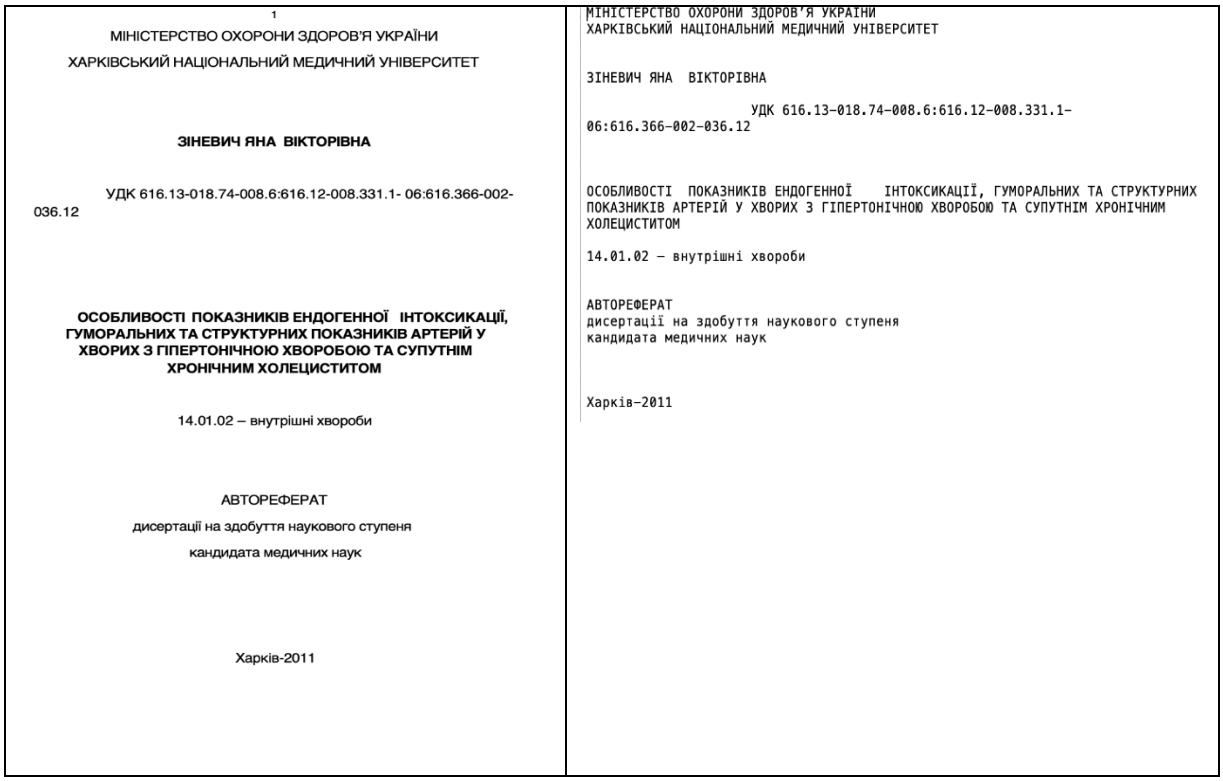
Рис. 1. Пример использования утилиты *textutil* для преобразования формата документа

На втором шаге подготовки документов, для того, чтобы ускорить синтаксический анализ, работающий недостаточно быстро для обработки целого документа, извлекается только титульная страница автореферата. Такой подход оказался допустимым, поскольку основная часть документа не содержит интересующих нас метаданных об автореферате. Эта информация находится на титульной странице либо на первых двух страницах автореферата. Титульную страницу извлекали из уже преобразованного простого текстового файла. Опытным путём было установлено, что 50 первых строк достаточно для извлечения всей необходимой информации и для существенного сокращения времени дальнейшей обработки. На рис. 2 показан пример команды для извлечения начальных строк из текстового файла.

```
head -n 50 file.txt > head.txt
```

Рис. 2. Пример использования утилиты *head* для усечения документа

На выходе после предварительной подготовки документа получаем файл, содержащий 50 строк текста, о которых известно, что в них записана титульная страница автореферата диссертации (см. рис. 3). Дальнейшая задача состоит в том, чтобы из этого небольшого текстового файла извлечь метаданные диссертации.



1

МІНІСТЕРСТВО ОХОРОНИ ЗДОРОВ'Я УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ МЕДИЧНИЙ УНІВЕРСИТЕТ

ЗІНЕВИЧ ЯНА ВІКТОРІВНА

УДК 616.13-018.74-008.6:616.12-008.331.1- 06:616.366-002-036.12

ОСОБЛИВОСТІ ПОКАЗНИКІВ ЕНДОГЕННОЇ ІНТОКСИКАЦІЇ, ГУМОРАЛЬНИХ ТА СТРУКТУРНИХ ПОКАЗНИКІВ АРТЕРІЙ У ХВОРИХ З ГІПЕРТОНІЧНОЮ ХВОРОБОЮ ТА СУПУТНІМ ХРОНІЧНИМ ХОЛЕЦИСТИТОМ

14.01.02 – внутрішні хвороби

АВТОРЕФЕРАТ

дисертації на здобуття наукового ступеня

кандидата медичних наук

Харків-2011

МІНІСТЕРСТВО ОХОРОНИ ЗДОРОВ'Я УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ МЕДИЧНИЙ УНІВЕРСИТЕТ

ЗІНЕВИЧ ЯНА ВІКТОРІВНА

УДК 616.13-018.74-008.6:616.12-008.331.1-06:616.366-002-036.12

ОСОБЛИВОСТІ ПОКАЗНИКІВ ЕНДОГЕННОЇ ІНТОКСИКАЦІЇ, ГУМОРАЛЬНИХ ТА СТРУКТУРНИХ ПОКАЗНИКІВ АРТЕРІЙ У ХВОРИХ З ГІПЕРТОНІЧНОЮ ХВОРОБОЮ ТА СУПУТНІМ ХРОНІЧНИМ ХОЛЕЦИСТИТОМ

14.01.02 – внутрішні хвороби

АВТОРЕФЕРАТ
дисертації на здобуття наукового ступеня
кандидата медичних наук

Харків-2011

Рис. 3. Титульный лист автореферата до и после предварительной подготовки

Минимальный набор возможных полей метаданных подлежащих извлечению:

- автор,
- тема,
- научная степень.
- раздел науки.

Дополнительные поля могут быть следующие:

- научный руководитель,
- оппоненты,
- организация,
- и другие.

3. Извлечение данных

Ранее нами уже был описан метод извлечения данных из слабоструктурированных текстов с помощью регулярных выражений [1]. В настоящей работе метод усовершенствован заменой регулярных выражений на полноценный синтаксический анализатор.

Известные существующие системы для лингвистического анализа произвольных русскоязычных текстов от Yandex[1] и Pullenti[2] представляют собой чёрные ящики с закрытым исходным кодом [2]. По этой причине они сложны в настройке и накладывают ограничения на программное окружение.

Сравнительно недавно был разработан лингвистический анализатор Yargy [3] с открытым исходным кодом на языке Python. Он позволяет выполнять синтаксический разбор текста на русском языке, что критически важно для наших целей, поскольку для английского языка существует множество решений, перенастройка которых на русский язык по сложности сравнима с написанием собственного лингвистического анализатора.

Анализатор Yargy изначально настроен на русский и, частично, украинский языки. Наша дополнительная задача доработать Yargy таким образом, чтобы он полноценно выполнял синтаксический разбор текстов на украинском языке. Эта задача упрощается тем, что в основе библиотеки Yargy есть встроенный морфологический модуль, который уже понимает, как русский, так и украинский язык. Благодаря этому модулю анализ украиноязычных текстов возможен «из коробки», однако одной лишь морфологии часто бывает недостаточно. Иногда происходят ошибки определения частей речи, иногда неправильно выделяются части слова. Например, мужское имя «Виктор» (рус.) и «Віктор» (укр.). В русском языке библиотека правильно определяет слово целиком как имя человека. В украинском языке выделяется приставка «вік» и корень «тор», тем самым препятствуя правильной классификации слова и затрудняя правильный анализ документа.

Рассмотрим библиотеку Yargy детальнее. Реализована библиотека целиком на языке Python, что позволяет запускать её везде, где можно использовать интерпретатор Python. На вход библиотека принимает, во-первых, специально подготовленную контекстно-свободную грамматику, настроенную на определённый тип документа в зависимости от задачи, и, во-вторых, собственно текст для анализа. На выходе работы анализатора получаем граф синтаксического разбора входного текста по данной грамматике. Таким образом, библиотека не содержит встроенную грамматику, что делает анализатор очень гибким и пригодным для разнообразных типов задач. Доступны два режима работы с входным текстом:

- найти все совпадения с грамматикой в тексте, то есть все подстроки, которые распознаются данной грамматикой;
- проверить, распознаётся ли весь заданный текст грамматикой.

Сама грамматика описывается также на чистом Python, что удобно, поскольку не требует установки различных сред и изучения дополнительного внутреннего языка. И хотя для описания грамматики авторами Yargy был разработан специальный язык DSL (domain specific language), он все же является подмножеством языка Python.

Синтаксический разбор реализован с помощью алгоритма Эрли [4]. Алгоритм

[1] https://yandex.ru/dev/tomita/
[2] www.pullenti.ru

кубической сложности, позволяющий разбирать произвольные тексты по контекстно-свободной грамматике, которая не требует приведения к нормальной первой форме Хомского. Контекстно-свободная грамматика, заданная на входе алгоритма, может содержать неоднозначности, что не будет препятствовать её разбору этим алгоритмом.

Дополнительным преимуществом при использовании Yargy является возможность добавить статистические анализаторы. Библиотеке на вход можно подать размеченные тексты, в которых лингвистические признаки слов такие как род, число, падеж, часть речи выписаны в последовательности вероятностных правил, что за чем следует и с какой вероятностью. Далее если в новом анализируемом тексте встречается незнакомое слово, которого нет в словарях, то библиотека может, основываясь на лингвистических признаках цепочки предыдущих слов, спрогнозировать род, падеж, число для незнакомого слова. Результаты такого рода обучения можно использовать как дополнительный вход библиотеки Yargy. В рамках настоящей работы статистический синтаксический анализ подробно не рассматривается.

Кроме синтаксического разбора библиотека Yargy позволяет размечать каждое слово в соответствии с его морфологией, то есть получать результаты морфологического анализа. Приписывание каждому слову род, число, падеж и часть речи помогает более детально настраивать дерево разбора. Используемый морфологический модуль pymorphy2 позволяет привести форму слова к нормальному виду (т. е. единственное число, именительный падеж для имен существительных, инфинитив для глаголов), что позволяет одновременно выполнять преобразования извлекаемых данных для сохранения сущностей в удобной форме. Например, если имя, фамилия и отчество человека встретилось в тексте в родительном падеже, то в базе данных они сохраняются в именительном падеже.

Модуль pymorphy2 основывает морфологический анализ на размеченном корпусе текстов русского языка OpenCorpora[3]. Поэтому морфологический анализ украинских слов не работает должным образом.

4. Грамматика

Для дальнейшей обработки подготовленных документов была построена грамматика, распознающая титульный лист автореферата диссертации.

Правила. На рис. 4 показан пример написания правил в этой грамматике.

```
NAME = rule(
   gram('NOUN')
)
PERSON = rule(
   SURN,
   NAME,
   PATR
)
```

Рис. 4. Фрагмент правила грамматики для извлечения имени из текста

В каждом автореферате диссертации присутствует автор. В приведенном примере правила указана фамилия SURN, имя NAME и отчество PATR. Именно в таком порядке. Каждое из этих слов — имя существительное. В грамматике указывается какой частью речи должно быть имя автора. В приведенном фрагменте имя NAME это имя существительное NOUN. В целом данное правило гласит, что если встречается подряд три существительных, то это полное имя человека, в данном случае имя автора автореферата. Это имя позднее будет извлечено для сохранения в базе и связи его с другими извлечёнными сущностями.

Также написаны отдельные правила для фамилии и отчества, описывающие дополнительные признаки для каждого части имени. Правила охватывают всевозможные сочетания для имён, но достаточно узко, чтобы не захватывать лишние данные. С помощью библиотеки синтаксического анализа Yargy построенная контекстно-свободная грамматика, применяющаяся к неструктурированному тексту

[3] http://opencorpora.org/

для получения на выходе структурированных данных, как в рассмотренном примере фамилию, имя и отчество автора. Если правила написаны достаточно хорошо, то эти данные определяются во входном документе. В противном случае мы получаем либо ложное срабатывание, когда грамматика либо её часть срабатывает не на тех данных, которые предполагалось извлечь с помощью этих правил, либо документ не соответствует ни одному варианту структуры документа, описанному грамматикой, и тогда распознавания не происходит вообще.

Для выделения имени человека существуют уже написанные более сложные правила. В частности, над рассматриваемой библиотекой Yargy[4] реализована использующая её библиотека Natasha[5], предназначенная для извлечения имен, адресов, организаций и некоторых других сущностей. Наборы правил из Natasha можно легко использовать в своих грамматиках для Yargy [3]. Для нашего корпуса текста грамматика распознавания русских имён из Natasha была доработана, поскольку в изначальном виде распознавание было успешным лишь на малом количестве тестовых документов.

Последовательность обработки текста при синтаксическом анализе следующая. Прежде всего весь текст разбивается на токены. Эта часть алгоритма предельно проста. Пробелы игнорируются, любая последовательность букв или цифр считается токеном, а также любой знак препинания тоже считается токеном. Такой подход позволяет писать правила, использующие знаки препинания для распознавания. Так можно распознавать специальные отформатированные данные, такие как УДК или даты.

Предикаты. В правилах можно использовать предикаты. Правило, показанное на рис. 4, использует предикат о том, что слово является существительным. На рис. 5 приводится правило для базового распознавания предметной классификации УДК.

Поле данных УДК представляет собой само слово “УДК”, за которым следуют цифры, точки, дефисы, двоеточия. В приведённом правиле используется два предиката eq и type для проверки полного совпадения и для проверки совпадения типа токена с целым числом либо со знаком препинания. Правило действует таким образом: сначала проверяется точное совпадение со словом “УДК”, затем с помощью логической связки or (логическое “или”) проверяется совпадение по одному из двух предикатов, что, либо тип токена это число, либо тип токена – знак пунктуации. Также можно указать, что правило является повторяемым, указав ключевое слово repeatable() для части правила.

```
UDK = rule(
   eq('УДК'),
   or_(
      type('INT'),
      type('PUNCT')
   ).repeatable()
)
```

Рис. 5. Фрагмент правила, распознающего предметную классификацию УДК

Морфология. Существует возможность проверять совпадения с любой формой проверяемого слова. Такая проверка, например, для слов 'кандидат' и 'доктор' осуществляется с помощью правила, показанного на рис. 6. Здесь идет проверка совпадения слов кандидата, кандидату, кандидаты, доктора, доктору и др. Для этого нужно либо выписать все словоформы, либо написать правило о том, чтобы находить совпадения в любом падеже, числе и роде. Либо использовать библиотечную функцию, упрощающую эту рутинную морфологическую работу.

```
DEGREE = rule(morph pipeline([
   'кандидат',
   'доктор'
]))
```

Рис. 6. Правило проверки морфологии слов 'кандидат' и 'доктор'

[4] https://github.com/natasha/yargy
[5] https://github.com/natasha/natasha

5. Интерпретация

После того как сделан разбор, и получено дерево синтаксического разбора текста, возникает задача привязки узлов дерева разбора к структурам данных. При использовании библиотеки Yargy для этой цели создаются объекты Python. Выше мы рассматривали пример простой грамматики для разбора полного имени человека. Для этой грамматики можно использовать объект Person на языке Python, показан на рис. 7. После анализа объект будет конкретизирован данными из результатов синтаксического разбора.

Привязка объекта к грамматике осуществляется с помощью так называемой интерпретации. К правилу грамматики приписывается объект, с помощью которого будет происходить интерпретация результатов работы этого правила. На рис. 8 показана реализация интерпретации правила распознавания имени объектом Person.

```
Person = fact(
   'Person',
   ['surn', 'name', 'patr']
)
```

Рис. 7. Объект Person

```
PERSON = rule(
   SURN,
   NAME,
   PATR
). interpretation(
   Person
)
```

Рис. 8. Интерпретация правила или привязка объекта к правилу

После составления дерева разбора, когда парсер распознает соответствующий текст, каждому интерпретируемому узлу дерева будет сопоставлен объект и распознанными текстовыми значениями будут инициализированы поля этого объекта. Каждое правило будет узлом либо листом в дереве разбора. Составные правила, которые включают в себя другие правила, будут узлом в дереве. Терминальные же символы и предикаты будут листьями такого дерева. Интерпретация узла будет объектом, а интерпретация листа будет атрибутом объекта. Например, значение имени будет атрибутом объекта Person.name.

6. Результаты разбора

На рис. 9 в формате JSON показан полный результат разбора титульной страницы автореферата.


```
{
  "author": {
    "surn": "Зіневич",
    "name": "Яна",
    "patr": "Вікторівна"
  },
  "udk": "УДК 616.13-018.74-008.6:616.12-
008.331.1- 06:616.366-002-036.12",
  "title": "ОСОБЛИВОСТІ ПОКАЗНИКІВ ЕН-
ДОГЕННОЇ ІНТОКСИКАЦІЇ, ГУМОРАЛЬ-
НИХ ТА СТРУКТУРНИХ ПОКАЗНИКІВ АР-
ТЕРІЙ У ХВОРИХ З ГІПЕРТОНІЧНОЮ
ХВОРОБОЮ ТА СУПУТНІМ ХРОНІЧНИМ
ХОЛЕЦИСТИТОМ",
  "specnum": "14.01.02",
  "specname": "внутрішні хвороби",
  "degree": "кандидат",
  "degreespec": "медичних наук",
  "city": "Харків",
  "year": "2011"
}
```


Рис. 9. Результат разбора титульной страницы автореферата в формате JSON

Далее на рис. 10 показана часть графа дерева разбора для Имени автора. Объект автор диссертации – Thesis.author, это персона Person, фамилия персоны Person.surn – ЗАЛОЗНОВА, имя персоны Person.narne – Юлія, отечество персоны Person.patr – Станіславівна. Это дерево разбора уже с привязанными объектами.

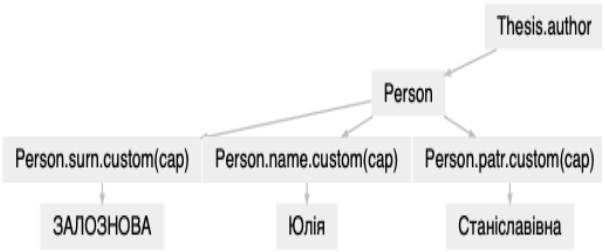


Рис. 10. Фрагмент дерева разбора

7. Дальнейшая работа

Сохранение и отображение данных. Следующая задача состоит в том, чтобы сохранить извлечённые результаты в базе данных для того, чтобы можно было делать запросы и представлять извлечённую информацию публично в удобочитаемом виде. Для хранения данных мы отдаём предпочтение формату RDF, поскольку в нём естественным образом поддерживается неограниченное количество атрибутов и связей, можно обрабатывать автоматически и есть возможность строить SPARQL-запросы.

Улучшение разбора. Предстоит существенно улучшить украинскую морфологию, а также добавить словари имен. Такие словари можно получить извлечением из названий статей Википедии, поскольку там уже есть большое количество имён, фамилий и отчеств, причём формат известен заранее. Извлечь их можно из соответствующих страниц категорий Википедии, где имена представлены списками на одной странице. Благодаря тому, что формат известен, можно разделить название статьи на имя, фамилию и отчество, и сохранить каждое поле в отдельный файл, получая таким образом словари имён, фамилий и отчеств. Эти словари необходимы для более точного распознавания имён. Если слово найдено в словаре, то распознавание такого слова в тексте становится тривиальной задачей с большой вероятностью успешного результата.

Выводы

В работе пошагово и на многочисленных примерах описан метод извлечения данных из слабоструктурированных текстов корпуса авторефератов диссертаций с помощью использования синтаксического анализатора Yargy.

В дальнейшем предстоит улучшить украинскую морфологию и добавить словари. А также разработать схему хранения извлеченных данных с тем, чтобы их можно было бы публично использовать и строить к ним различные запросы.

Работа выполняется в рамках темы "Методы и средства создания интеллектуальных сервис-ориентированных информационно-обеспечивающих систем в среде семантического Веба".

Литература

1. Кудим К.А., Проскудина Г.Ю. Методы и средства извлечения данных о персоналиях из авторефератов диссертаций. *Проблеми програмування*. 2019. № 2. С. 38–46.
2. Рубайло А.В., Косенко М.Ю. Программные средства извлечения информации из текстов на естественном языке. *Альманах современной науки и образования.* № 12 (114). 2016. С. 87–92. http://scjournal.ru/articles/issn_1993-5552_2016_12_23.pdf
3. Кукушкин А. Наташа – библиотека для извлечения структурированной информации из текстов на русском языке. https://habr.com/ru/post/349864/
4. Earley J. An efficient context-free parsing algorithm, Communications of the Association for Computing Machinery. 13:2:94-102. 1970.

References

1. Kudim K.A., Proskudina G.YU. (2019). Methods and tools for extracting personal data from theses abstracts *Problems in programming*. [online – pp.isofts.kiev.ua] (2). P. 38–46. (in Russian). Available from: http://pp.isofts.kiev.ua/ojs1/article/view/359 [Accessed 6/05/2019]. DOI: https://doi.org/10.15407/pp2019.02.038
2. Rubailo A.V., Kosenko M.Y. Software tools for information extraction from natural-language texts. *Almanac of modern science and education*. № 12 (114) 2016. P. 87–92. (in Russian). http://scjournal.ru/articles/issn_1993-5552_2016_12_23.pdf
3. Kukushkin A. Natasha – a library for extracting structured information from texts in Russian. (in Russian). https://habr.com/ru/post/349864/
4. Earley J. An efficient context-free parsing algorithm, Communications of the Association for Computing Machinery, 13:2:94-102, 1970.

Получено 03.02.2020

***Об авторах*:**

Кудим Кузьма Алексеевич,
младший научный сотрудник.
Количество научных публикаций в украинских изданиях – 18.
Количество научных публикаций в зарубежных изданиях – 2.
http://orcid.org/0000-0001-9483-5495,

Проскудина Галина Юрьевна,
научный сотрудник.
Количество научных публикаций в украинских изданиях – 31.
Количество научных публикаций в зарубежных изданиях – 15.
http://orcid.org/0000-0001-9094-1565.

Место работы авторов:

Институт программных систем
НАН Украины,
03187, Киев-187,
проспект Академика Глушкова, 40.
Тел.: (044) 526 6033.

E-mail: kuzmaka@gmail.com,
guproskudina@gmail.com