УДК 004.42:81-13 DOI: 10.15593/2224-9389/2017.2.7

Р.Д. Каримов, А.А. Акинин

Челябинский государственный университет,
Челябинск, Российская Федерация

Получена: 01.05.2017
Принята: 08.05.2017
Опубликована: 30.06.2017

РАСШИРЕННЫЙ РЕДАКТОР КОРПУСА С ГРАФИЧЕСКИМ ПОЛЬЗОВАТЕЛЬСКИМ ИНТЕРФЕЙСОМ

Рассматривается приложение для ручного создания, редактирования и аннотирования корпуса текстов. Приложение снабжено графическим пользовательским интерфейсом и предназначено для частично автоматизированного построения корпуса, осуществляемого посредством автоматического заполнения разметки лексического уровня на основе предыдущего пользовательского ввода. В отличие от некоторых имеющихся корпусных приложений с графическим интерфейсом данная разработка не привязана функционально к конкретному корпусу или языку, может быть использована лингвистами без навыков программирования и предлагает более широкий функционал лексикограмматического аннотирования. На текущем этапе разработки предварительно настроена структура разметки, основанная на лейпцигской грамматической нотации. Сохранение разметки непосредственно в текстовом файле, содержащем корпус, вкупе с применением распространенных и интуитивно понятных грамматических обозначений обеспечивает простую человекочитаемость созданных или обработанных в данном приложении текстовых ресурсов, а также возможность их конвертации для последующего использования в других приложениях. Пользовательский ввод первоочередно осуществляется вручную и иерархизирован, т.е. заполнение разметки производится дедуктивно от наиболее общих признаков той или иной лексемы к ее частным свойствам, при этом записанные пользователем данные в форме тегов в последующем обеспечивают частичное автозаполнение грамматических свойств в разметочном слое. Программа в первую очередь предназначена для упрощения работы с небольшим корпусом текстов. Планируется расширить и оптимизировать ее функционал за счет доработки генерируемых индексных файлов; автоматическое аннотирование может быть улучшено путем применения векторных моделей лексической близости.

Ключевые слова: *корпус, графический интерфейс, автоматизированное аннотирование, разметка, автозаполнение.*

R.D. Karimov, A.A. Akinin

Chelyabinsk State University,
Chelyabinsk, Russian Federation

Received: 01.05.2017
Accepted: 08.05.2017
Published: 30.06.2017

ADVANCED CORPUS EDITOR WITH A GRAPHICAL USER INTERFACE

The paper describes an application for manual creation, editing, and annotation of text corpora. The app has a graphical user interface (GUI) and is designed for computer-aided corpus building by means of auto-propagation of lexical mark-up based on previous user input. Unlike some of the existing corpus apps with GUI, this development is not functionally bound to any specific corpora or languages; it can be used by linguists unskilled at programming and offers a broader range of lexicogrammatical annotation functions. At the current stage of development, we have configured a markup architecture based on Leip-

zig glossing rules; markup is saved directly in the plain text corpus files. Coupled with the well-known and intuitive grammatical notation, it contributes to the human-readability of the text resources created or processed by the app while also making them easily convertible for use with other applications. User input is primarily manual and follows a deductive hierarchy, i.e. the user first specifies more general properties of a lexeme, then more specific ones. The tag-like user-entered data further enable partial auto-propagation of grammatical properties in the markup layer. The editor is primarily intended to facilitate operations with small corpora. It is planned to expand and optimize its functions by improving the generation of index files; automatic annotation can be enhanced by use of vector space models.

Keywords: *corpus, GUI, computer-aided annotation, markup, auto-propagation.*

Введение. В настоящее время практически отсутствует инструментарий для работы с корпусом, пользовательское взаимодействие с которым осуществлялось бы через графический интерфейс. В то же время можно отметить такие программы, как Atomic [1] и Gate (General Architecture for Text Engineering) [2], но ни одна не является решением, подходящим для переводчиков и филологов, не обладающих обширными знаниями в корпусной и прикладной лингвистике: первая предназначена для графического обозначения связей между элементами текста, вторая является средой для упрощенной разработки и последующего запуска NLP-приложений. Открытость исходного кода данных платформ предполагает возможность написания пользовательских модулей, однако, на наш взгляд, для лингвистов, не специализирующихся в NLP (компьютерной обработке естественного языка), важны инструменты, позволяющие обрабатывать текст без взаимодействия с компьютерным кодом или командной строкой. Таким приложением является разработанная К.А. Дрогановой и Н.С. Медянкиным программа с упрощенным веб-интерфейсом, осуществляющая морфологическое и синтаксическое аннотирование текстов на русском языке [3]; однако как лингвистическая ограниченность, так и невозможность пользовательской настройки самой разметки (без редактирования исходных кодов) являются, на наш взгляд, существенными недостатками.

Настоящее исследование посвящено разработке приложения для создания, редактирования и аннотирования корпусов текста с гибкой архитектурой разметки, настраиваемой под решение конкретных задач, для которых создается корпус.

Постановка задачи. В рамках исследования текста, а также извлечения определенных лингвистических данных, включая данные о частотности употребления лексем или каких-либо морфологических/синтаксических конструктов, необходимо создание структурированного корпуса с комплексной разметкой и архитектурой, обеспечивающей возможность расширенного поиска по тексту. В частности, подобные функции обеспечивают возможность исторического исследования статистических изменений в составе словаря [4], а также позволяют выявлять лексикограмматические тенденции, что дает возможность, например, отслеживать стабилизацию глагольного спряжения за счет тематизации [5]. В соответствии с этим перед разработчиками приложения для создания корпуса и работы с ним ставятся задачи по реализации следующего функционала:

- выделение словарных и инфлектированных форм, сбор данных по их частотности;
- отслеживание характера употребления специфических форм и синтагм;
- определение и моделирование орфографических, морфографемических, морфофонетических и иных особенностей различных словоупотреблений.

Такая архитектура предполагает дополнение текста обширным слоем разметки с целью описания различных свойств как парадигматического, так и синтагматического уровня, при этом такая разметка может сопровождать основной текст как в самом текстовом файле (как это реализовано в [3]), так и во вспомогательном файле-индексе. В то время как для распространенных языков с большими текстовыми базами имеются уже реализованные корпусы с расширенной разметкой, в том числе НКРЯ, далеко не всегда программное обеспечение, предназначенное для работы с данными корпусами, позволяет вносить пользовательские изменения в текст корпуса, дополнять его, а также изменять сам характер разметки под нужды пользователя. Кроме того, функционал веб-интерфейса таких программ зачастую не предоставляет возможности сбора статистических данных. Проблема становится еще более явной при работе с историческими или некоторыми малораспространенными языками/диалектами, для которых не существует готовых корпусов. В связи с этим исследователями была сформулирована задача по разработке приложения со следующими функциями:

1) создание и/или импорт обычного текстового файла;

2) возможность добавления пользовательской разметки на уровне отдельных слововхождений для описания морфологических, синтаксических или иных свойств, являющихся лексема-специфичными;

3) возможность добавления метатегов для описания крупных сегментных (словосочетание, предложение) и супрасегментных единиц (текстов или текстовых фрагментов с общей дискурсивной, интенциональной или стилистической базой;

4) автоматическое заполнение разметки на основе предыдущего пользовательского ввода;

5) корпус внутри поиска и сбор статистических данных, в том числе на основе тегированного поиска;

6) самостоятельная пользовательская настройка разметки, включая конфигурирование тегов и формата выходных файлов программы;

7) ведение корпусного проекта, включающего файл корпуса, файл базы данных, а также конфигурационные файлы.

Подобные функции должны быть доступны через интуитивно понятный графический пользовательский интерфейс и разделены на две группы: первая представлена в основном окне программы, вторая – в разделе ее настроек.

Текущее состояние. Исследователями ранее было реализовано приложение на языке Python, осуществляющее лемматизацию текста на средне-

вековом английском с возможностью конфигурирования под иные языки со схожей грамматической структурой [6]. Приложение использует три типа входных файлов: один простой текстовый файл с подлежащим лемматизации текстом, один табличный файл, перечисляющий возможные морфы с указанием их частеречной принадлежности, и табличные файлы, содержащие лексикон – словарь лемм и возможных орфографических вариаций слова. Конфигурирование программы таким образом осуществляется через простое изменение входных файлов: словаря морфов и лексикона лемм. Работа алгоритма основывается на простом усечении токенизированных слов из корпуса и последующем сопоставлении частеречной принадлежности полученных таким усечением морфов и лемм, найденных простым поиском по лексикону. Проведенная с помощью данного инструмента лемматизация одного из диахронических разделов Хельсинкского корпуса английских текстов [7] позволила значительно повысить статистическую репрезентативность данного корпуса, что было оценено и подтверждено по закону Ципфа и проверке его выполнения по критерию Пирсона («хи-квадрат»):

Сравнение некоторых статистических
показателей текста до и после лемматизации

Показатель	До лемматизации	После лемматизации	Комментарий
s-экспонента по закону Ципфа	0,8546697	0,9625564	Рассчитана с помощью пакета lzipf языка программирования R
Хи-квадрат	68,5504	26,52005	Взято 18 степеней свободы, для которых при доверительном интервале квантиль распределения равняется 28,8693 [8].

При расчете показателя хи-квадрат сравнивали фактическое распределение частотностей до и после лемматизации с распределением, рассчитанным по закону Ципфа с соответствующим значением s. Превышение квантиля распределения считалось признаком несоответствия фактического распределения гипотетическому; таким образом пришли к выводу, что только в результате лемматизации корпуса текста полученным инструментом было обеспечено выполнение закона Ципфа.

Хотя описанное приложение при относительной простоте использования позволяет реализовывать одну из задач построения корпусной архитектуры – получение лемматического списка, эффективность его работы зависит от корректности составляемых морфологических словарей и лексикона лемм, что делает предварительную подготовку чрезмерно трудоемкой при необходимости обработать большой корпус. Применение словаря было необходимо ввиду того, что лемматизирующий скрипт создавался для обработки текстов

на средневековом английском, отличающимся значительной варьируемостью орфографии, обусловленной, помимо прочего, взаимозаменяемостью отдельных графем и графемических кластеров. Так, кластер *th* стал заменять буквы Þ и Ð в написании, но данное изменение в течение долгого времени не было повсеместным [9]. Некоторые иные существующие на сегодняшний день алгоритмы лемматизации также базируются на предварительно составляемых табличных выборках и могут обучаться путем генерации двухуровневых правил на основе сопоставления морфографемики словарных и инфлектированных форм, заранее соотнесенных пользователем [10]. В рамках такого подхода применяется машинное обучение с помощью конечных преобразователей [11], что, на наш взгляд, также малоприменимо при работе с небольшим корпусом текстов. Кроме того, лемматизация является лишь одной из обширного списка задач корпусной лингвистики, и даже ее выполнение данным алгоритмом сопряжено с рядом трудностей, обусловленных плохой конфигурируемостью и масштабируемостью подхода, основанного на усечении морфов и словарном поиске. В связи с этим было решено фактически обратить процесс работы с корпуса – реализовать автоматическую генерацию лексикона и морфологического слоя на основе пользовательского ввода.

В данный момент ведется разработка приложения на языке Python, выбор которого обусловлен тем, что в нем предоставлены проработанные высокоуровневые структуры данных и простой эффективный подход к объектно-ориентированному программированию. Кроме того, Python [12] является идеальным языком для написания сценариев и ускоренной разработки приложений в различных сферах и на большинстве платформ. К настоящему времени для приложения реализованы основное окно и обработчик текста, осуществляющий пословную и сентенциальную сегментацию по упрощенным правилам:

– пробел маркирует границу слова;

– точка, восклицательный или вопросительный знак маркируют границу предложения;

– прочие символы, не являющиеся буквами или дефисом, игнорируются.

После предварительной сегментации текста пользователь может выделять отдельные слова и указывать их свойства в выпадающем списке. Предварительно настроенная конфигурация разметки ориентирована на германские языки; сами свойства иерархизированы, при этом вершиной иерархии является частеречная принадлежность. Поэтому для каждого слова пользователь сначала указывает часть речи, затем получает возможность выбрать грамматические свойства, специфичные для нее (вид и время – только у глаголов, род и падеж – только у именных частей речи и т.д.). По завершении работы со словом пользователь сохраняет его и переходит к следующему, при этом при нахождении программой в тексте нескольких полностью графически совпадающих с размеченной лексемой слововхождений пользова-

тельский интерфейс предлагает автоматически применить сохраненную разметку к таким совпадениям.

Сохранение самой разметки производится в несколько файлов.

Основной файл корпуса содержит непосредственно текст, где после каждого слова вводятся в виде тегов приписанные ему свойства, при этом в качестве тегов используются аббревиатуры лейпцигской грамматической нотации [13], что, на наш взгляд, обеспечивает лучшую человеко-читаемость выходного файла, а также его пригодность к конвертированию для использования с другими корпусными инструментами. В конце каждого предложения указываются его свойства, при этом для выделения тегов сентенциального уровня используются другие делимитаторы; кроме этого для облегчения индексации каждому предложению приписывается его порядковый идентификационный номер в корпусе. Сам текст в файле выглядит следующим образом:

I /PRON,1SG,lemma=I/ *am* /V,AUX,PRS,IND,1SG,lemma=be/ [narr,non-exclam,ID=XXX].

Файл словоформ, формат которого также определен предварительной конфигурацией, является табличным и содержит несколько листов, каждый из которых резервируется под отдельную часть речи. Левый столбец предназначен для записи леммы, второй – для сохранения неизменяемых характеристик (род существительного, глагольный класс и т.д.), третий и последующие – для фиксации различных форм, которые пользователь соотнес с данной леммой. Таким образом, по ходу работы с корпусом программа генерирует табличный лексикон.

Третий файл является синтаксическим индексом, в котором фиксируются синтагматические конструкты, выявленные в тексте на уровне предложений на основе пользовательского ввода. Формат записи примерно соответствует синтаксическому разбору, используемому в некоторых контекст-независимых грамматиках, например:

$$S \rightarrow NP(Adj+N)VP(Aux+NP(Adj))\ (114,ID=XXX,YYY,ZZZ),$$

где цифра в скобках справа показывает, сколько раз данный конструкт был определен в уже обработанном тексте, а после нее перечисляются идентификаторы предложений, соответствующих описанной конструкции.

На данный момент функция автозаполнения разметки лексического уровня реализована лишь частично и основывается на посимвольном сопоставлении графической реализации лексем с формами, сохраненными в табличном файле; при переходе к следующему токену программа выбирает из лексикона токены, символьный состав которых на 80 % и более совпадает с составом анализируемого слова, и предлагает автоматическое заполнение леммы и части тегов из табличной базы, при этом перечень предлагаемых вариантов сужается по мере заполнения разметки. Например, если в лексиконе сохранены слова *to bring* (V) и *bringer* (N), то при переходе к токену brings программа предложит оба вариан-

та сохраненной разметки, но после указания части речи – только глагольную разметку. После указания частеречной принадлежности хотя бы первого слова в предложении программа предлагает пользователю выбрать синтаксический конструкт из индекса и автоматически заполняет значения тега части речи для остальных слов в предложении на основе такого выбора.

Выводы и дальнейшие перспективы. Таким образом к настоящему моменту реализовано приложение, выполняющее некоторые базовые функции создания, редактирования, ручного и частично автоматизированного аннотирования корпуса текстов; последняя операция базируется на генерации табличных лексиконов и синтаксических индексов на основе пользовательского ввода. Предварительная конфигурация разметки, формата лексикона и индекса ориентирована на германские языки, но в дальнейшем предполагается возможность самостоятельной настройки данных функций со стороны пользователя.

Слабостью программы в ее текущей реализации является недостаточность представленных функций автоматической генерации разметки. В дополнение к уже описанному считаем необходимым разработать алгоритм полиграммного поиска с помощью синтаксического индекса с доработкой самого формата индексного файла; планируется включение в его записи полной конструкции предложений (с указанием падежных, числовых и иных характеристик отдельных элементов таких конструктов) для улучшения качества автогенерации тегов при обращении к индексу. Кроме того, рассматривается возможность предположения лемм на базе векторной близости [14]. Применение нейронных сетей в данный момент не представляется целесообразным, так как программа в первую очередь нацелена на небольшие корпусы, создаваемые пользователем для личных целей, что не обеспечит достаточность выборки для эффективного обучения нейросетевых моделей.

Основной из пока нерешенных задач является реализация функций поиска по корпусу, KWIC-представления и сбора статистических данных; ожидается, что разработка некоторых из этих функций не вызовет значительных затруднений, так как они уже реализованы в различных приложениях с открытым исходным кодом, как, например, KWIC – де-факто стандарт программ конкордантного представления и поиска [15].

Список литературы

1. Atomic [Электронный ресурс]. – URL: http://corpus-tools.org/atomic/index.html (дата обращения: 29.04.2017).

2. General architecture for text engineering [Электронный ресурс]. – URL: https://gate.ac.uk (дата обращения: 27.04.2017).

3. Droganova K.A., Medyankin N.S. NLP pipeline for Russian: an easy-to-use web application for morphological and syntactic annotation, Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference “Dialogue 2016”,

Moscow [Электронный ресурс]. – URL: http://www.dialog-21.ru/media/3451/droganovakamedyankinns.pdf (дата обращения: 20.04.2017).

4. Арапов М.В., Херц М.М. Математические методы в исторической лингвистике. – М.: Наука, 1974.

5. Николаева Н.А. Тематизация презенса сильного глагола в кельтских и германских языках (на материале древнеирландского и готского): автореф. дис. … канд. филол. наук. – М., 2003.

6. Акинин А.А., Каримов Р.Д., Никитина С.А., Самкова М.А. Свидетельство о государственной регистрации программы для ЭВМ № 2016615193 MiddleEnglishLem, 17.05.2016.

7. The Helsinki Corpus of English Texts [Электронный ресурс]. – URL: http://clu.uni.no/icame/hc/index.htm (дата обращения: 10.04.2017).

8. Квантили распределения «хи-квадрат» [Электронный ресурс]. – URL: http://ru.wikipedia.org/?oldid=80428674 (дата обращения: 30.04.2017).

9. Ward A.W., Waller A.R. Changes in the Language to the Days of Chaucer: Middle English Spelling, in The Cambridge History of English and American Literature: in 18 vol. Vol. I. From the Beginnings to the Cycles of Romance. – Cambridge: Cambridge University Press, 1907.

10. Jongejan B., Dalianis H. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike // Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP. – 2007. – P. 145–153.

11. Beesley K.R., Karttunen L. Finite State Morphology // Journal of Computational Linguistics. – 2004. – Vol. 30, № 2. – P. 237–249.

12. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012.

13. The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses [Электронный ресурс]. – URL: http://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf.

14. Turney P.D., Pantel P. From Frequency to Meaning: Vector Space Models of Semantics // Journal of Artificial Intelligence Research. – 2010. – № 37. – С. 141–188.

15. Kilgariff A., Kosem I. Corpus tools for lexicographers [Электронный ресурс]. – URL: https://www.sketchengine.co.uk/documentation/raw-attachment/wiki/AK/Papers/2011-KilgKosem-eLexbk.pdf.

References

1. Atomic, available at: http://corpus-tools.org/atomic/index.html (accessed 29 April 2017).

2. General architecture for text engineering, available at: https://gate.ac.uk (accessed 27 April 2017).

3. Droganova K.A., Medyankin N.S. NLP pipeline for Russian: an easy-to-use web application for morphological and syntactic annotation. *Computational Linguistics and Intellectual Technologies. Proceedings of the International Conference "Dialogue 2016"*, available at: http://www.dialog-21.ru/media/3451/dro-ganovakamedyankinns.pdf (accessed 20 April 2017).

4. Arapov M.V., Kherts M.M. Matematicheskie metody v istoricheskoi lingvistike [Mathematic methods in histrorical linguistics]. Moscow, Nauka, 1974.

5. Nikolaeva N.A. Tematizatsiia prezensa sil'nogo glagola v kel'tskikh i germanskikh iazykakh (na materiale drevneirlandskogo i gotskogo) [Thematisation of present tense of strong verbs in Celtic and German languages (by the example of Old Norse and Gothic)]. Abstract of Ph. D. thesis. Moscow, 2003.

6. Akinin A.A., Karimov R.D., Nikitina S.A., Samkova M.A. Svidetel'stvo o gosudarstvennoi registratsii programmy dlia EVM № 2016615193 MiddleEnglishLem [Certificate of the state registration of the computer application MiddleEnglishLem, no. 2016615193], 17 May 2016.

7. The Helsinki corpus of English texts, available at: http://clu.uni.no/icame/hc/index.htm (accessed 10 April 2017).

8. Kvantili raspredeleniia "khi-kvadrat" [Inverted distribution of chi square], available at: http://ru.wikipedia.org/?oldid=80428674 (accessed 30 April 2017).

9. Ward A.W., Waller A.R. Changes in the language to the days of Chaucer: Middle English spelling. *The Cambridge history of English and American literature*. Vol. I. From the beginnings to the cycles of romance. Cambridge, Cambridge University Press, 1907.

10. Jongejan B., Dalianis H. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, 2007, pp. 145-153.

11. Beesley K.R., Karttunen L. Finite state morphology. *Journal of Computational Linguistics*, 2004, vol. 30, no. 2, pp. 237-249.

12. Prokhorenok N.A. Python 3 i PyQt. Razrabotka prilozhenii [Python 3 and PyQt. Building applications]. Saint-Petersburg, BKhV-Peterburg, 2012.

13. The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses, available at: http://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf (accessed 20 April 2017).

14. Turney P.D., Pantel P. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 2010, no. 37, pp. 141-188.

15. Kilgariff A., Kosem I. Corpus tools for lexicographers, available at: https://www.sketchengine.co.uk/documentation/raw-attachment/wiki/AK/Papers/2011-KilgKosem-eLexbk.pdf (accessed 20 April 2017).

Сведения об авторах

КАРИМОВ Рауль Дамирович
e-mail: *raoul.karimov@hotmail.com*

Аспирант кафедры романо-германских языков и межкультурной коммуникации, Челябинский государственный университет (Челябинск, Россия)

About the authors

Raul D. KARIMOV
e-mail: *raoul.karimov@hotmail.com*

Postgraduate Student, Department of Romance-Germanic Languages and Cross-cultural Communication, Chelyabinsk State University (Chelyabinsk, Russian Federation)

АКИНИН Андрей Анатольевич
e-mail: *akinin96@gmail.com*

Кафедра теории управления и оптимизации, Челябинский государственный университет (Челябинск, Россия)

Andrey A. AKININ
e-mail: *akinin96@gmail.com*

Undergraduate Student, Department of Control and Optimization Theory, Chelyabinsk State University (Chelyabinsk, Russian Federation)