В. И. Керножицкая
Минск, МГЛУ

ПРОТОТИПИРОВАНИЕ ЛИНГВИСТИЧЕСКИХ ПРОЦЕССОРОВ ДЛЯ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ ЗАДАЧ АВТОМАТИЧЕСКОГО АНАЛИЗА ТЕКСТА

В работе подчеркивается актуальность задачи базового лингвистического анализа текстовой информации и лингвистического процессора как основного компонента ее эффективного решения. Излагается актуальность построения лингвистического процессора на основе инструментальных средств, находящихся в свободном доступе. Приводится описание этих средств и принципиальная схема алгоритма решения задачи базового лингвистического анализа текста с использованием такого типа лингвистического процессора.

Решение актуальных задач автоматической обработки текста, как правило, требует функциональности базового лингвистического анализа (БЛА) текста, который включает его преформатирование и распознавание границ слов и предложений, лексико-грамматический, синтаксический и семантико-синтаксический анализ входного текста [1].

Как показал проведенный анализ, на настоящее время существует многоязычный лингвистический процессор (ЛП) «IHS GoldFire» [Там же], являющийся одним из лучших среди такого класса разработок, а также достаточно эффективно решающий задачу БЛА текста. Заметим, что эта эффективность обеспечивается здесь развитой лингвистической базой знаний, которая включает такие языковые ресурсы, как большого объема словари словоформ, тегированные лексико-грамматическими классами слов, тегированные корпуса текстов, множество лингвистических паттернов автоматического синтаксического и семантико-синтаксического анализа текста и специальные языки их формального описания, вероятностные модели лексико-грамматического анализа текста, специальные словари, классификаторы и т.п. К тому же, здесь уделено самое пристальное внимание эффективности алгоритмического обеспечения ЛП, которое в своем ядре основывается на теории конечных автоматов.

К сожалению, данный ЛП не находится в свободном доступе, поэтому выходом из данной ситуации может быть либо его полная самостоятельная разработка, что чрезвычайно трудоемко, либо попытка построения ЛП из готовых, находящихся в свободном доступе модулей, решающих отдельные подзадачи БЛА текста. Второй вариант является более предпочтительным, особенно в тех случаях, когда необходимо достаточно оперативно построить прототип найденного возможного решения исследуемой задачи автоматической обработки текста и оценить в принципе его достоинства и перспективу,

не заостряя при этом особого внимания на качественных характеристиках, которые в будущем, в случае положительного мнения, очевидно, могут быть существенно улучшены. Именно о таком варианте построения ЛП идет речь в нашем случае.

В результате анализа требований, предъявляемых к лингвистическому процессору рассматриваемого типа и существующих для их удовлетворения программных решений, было решено включить в состав предлагаемой технологии построения требуемого ЛП следующие компоненты: Java, в качестве языка программирования для реализации ЛП, библиотеку Stanford CoreNLP [2], библиотеку Stanford Dependency Parser [3].

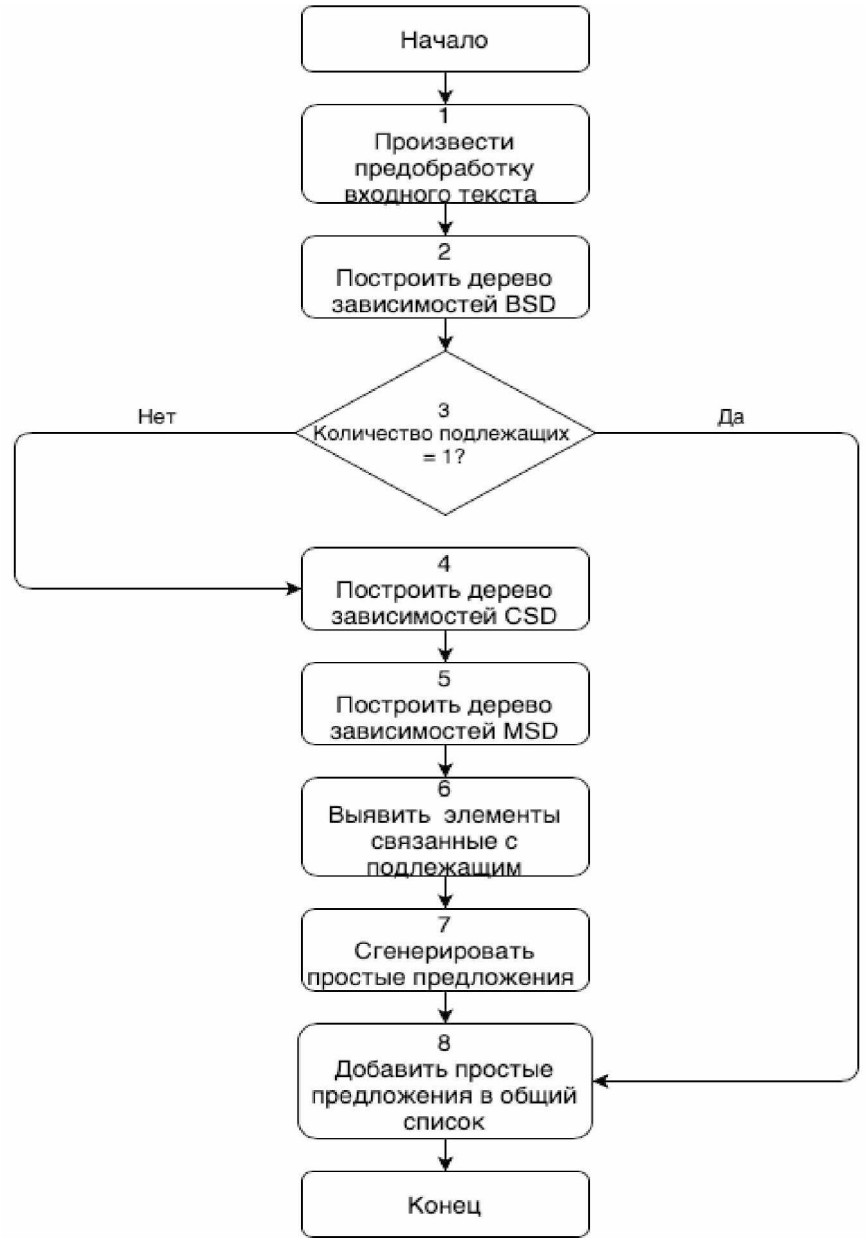
Stanford CoreNLP – библиотека, предоставляющая набор инструментов для обработки текста, основанная на работах стэнфордской группы обработки естественного языка (The Stanford Natural Language Processing Group). Работы коллектива ведутся как в фундаментальных областях компьютерной лингвистики, так и в ее прикладных аспектах: понимание предложений, машинный перевод, вероятностный парсинг и лингвистическая разметка, информационный поиск, снятие смысловой неоднозначности и т.д. Данные исследования в основном связаны с поддержкой английского языка, но также поддерживают арабский, китайский, французский и немецкий языки. Что касается непосредственно функционала разработанной коллективом библиотеки, то он включает все инструменты, необходимые для эффективной обработки текста, а именно: частеречную разметку, токенизацию текста, парсинг и т.д.

Stanford Dependency Parser – библиотека, предоставляющая инструменты для автоматического анализа грамматической структуры предложения в рамках грамматики зависимостей. В частности, она включает функционал для анализа грамматической структуры предложения, устанавливая связи между отдельными его составляющими и тегируя каждую выявленную связь типом грамматических отношений, которые эта связь иллюстрирует. Результаты анализа предложения представляются в виде дерева зависимостей, тип которого возможно, что очень важно, настраивать в зависимости от решаемой задачи. Типы деревьев зависимостей, согласно источнику [4], включают в себя Basic Stanford Dependency (BSD) и Collapsed Stanford Dependency (CSD). Полученные в результате работы библиотеки структуры затем доступны для дальнейшего анализа.

В целом имеет место следующая принципиальная схема алгоритма решения задачи БЛА текста [5] (рисунок).

Блок 1 осуществляет токенизацию входных данных, а именно, разбивает текст на отдельные предложения. На данном этапе также проводится частеречная разметка полученных предложений, из которых затем удаляются стоп-слова. Полученные в результате такого анализа предложения доступны для дальнейшей обработки. Для реализации этого этапа используются инструменты библиотеки Stanford CoreNLP.

Блоки 2–8 реализуют алгоритм автоматической генерации простых предложений из сложных. Необходимость данного этапа обработки текста заключается в том, что именно сложные предложения являются своего рода препятствием для повышения эффективности последующих этапов БЛА входного текста.



Принципиальная схема алгоритма решения задачи БЛА текста

Блок 2, в свою очередь, с использованием инструмента Stanford Dependency Parser обеспечивает построение дерева зависимостей «Basic Stanford Dependency», для входного предложения, которое затем доступно для дальнейшего анализа.

В блоке 3 происходит выявление сложных предложений и их отделение от простых, для чего проводится анализ дерева зависимостей, сгенерированного в блоке 2. Анализ заключается в подсчете количества присутствующих в дереве зависимостей входного предложения подлежащих, которые, согласно

«Stanford Typed Dependency Manual» [4], обозначаются тегами «nsubj» или «nsubjpass».

Очевидно, что в простом предложении будет только одно подлежащее с тегом «nsubj» или «nsubjpass», и, следовательно, предложение с количеством «nsubj» или «nsubjpass», превышающим 1, не будет считаться простым.

Построенное в блоке 4 дерево зависимостей CSD далее используется в блоке 5, являясь, наряду с BSD, основой для построения дерева зависимостей «Modified Stanford Dependency» (MSD).

Таким образом, блок 5 обеспечивает построение дерева зависимостей типа MSD, которое, в свою очередь, основывается на деревьях зависимостей типа BSD и CSD. А именно, MSD создается путем извлечения подлежащих с тегами «nsubj» или «nsubjpass» из CSD, а затем выявления и поочередного включения из BSD в MSD всех остальных элементов, связанных с каждым из выявленных подлежащих. Важно отметить, что, для улучшения качества дальнейшей обработки данных, полученных в результате БЛА при построении MSD, в нее не включают служебные части речи.

В блоке 6 из зависимости MSD для каждого подлежащего, т.е. главного слова, извлекается связанное с ним тегом «nsubj» или «nsubjpass» зависимое слово. Далее происходит поиск в MSD уникальных элементов дерева зависимостей, связанных с главным словом, а затем связанных с зависимым словом.

Каждому найденному элементу присваивается его порядковый номер в исходном предложении, по которому затем в блоке 7 восстанавливается структура и порядок слов генерируемых простых предложений. Блок 7 также отвечает за формирование списка из сгенерированных простых предложений, который затем в блоке 8 добавляется в общий список изначально простых предложений.

Таким образом, в качестве результата БЛА текста на выходе имеется сгенерированный из исходного текста набор простых предложений, для каждого словоупотребления которых зафиксированы как лексико-грамматические классы, так и типы грамматических отношений, которыми связаны отдельные словоупотребления входного текста.

Отметим, что предложенная модель лингвистического процессора БЛА текста, при решении конкретных задач его автоматической обработки, потребует, очевидно, расширения функциональности ЛП. Речь, прежде всего, идет о возможности поддержки нескольких языков (так называемая функциональность multi-language). Это вполне реализуемо – проведенный анализ показал, что на сегодняшний день существует достаточно большое количество соответствующих инструментальных средств. Также в дальнейшем неизбежно встанет вопрос о повышении качественных характеристик ЛП. Так, например, в предложенном варианте с этой целью на лексическом уровне используется словарь стоп-слов. Очевидно, что при необходимости он может быть дополнен списком наиболее частотных вводных и других слабо информативных конструкций. Однако такое решение имеет существенные ограничения, преодоление которых потребует использования известного механизма рас-

познающих лингвистических паттернов, но и это является возможным, благодаря широкому набору находящихся в свободном доступе соответствующих инструментальных средств.

В целом, исследования показали целесообразность использования представленного выше прототипа лингвистического процессора, по крайней мере, при экспериментальных исследованиях различных задач автоматической обработки текста.

ЛИТЕРАТУРА

1. *Чеусов, А. В.* Разработка алгоритмов и технологии построения многоязычного базового лингвистического процессора : дис. … канд. технич. наук : 05.13.17 / А. В. Чеусов. – Минск, 2013. – Л. 1–116.
2. Stanford CoreNLP – Natural Language Software [Electronic resource]. – Stanford, California, 2020. – Mode of access : https://stanfordnlp.github.io/CoreNLP/. – Date of access : 01.03.2020.
3. *Marneffe, M. C.* Generating Typed Dependency Parses from Phrase Structure Parses / M. C. Marneffe, B. MacCartney, C. Manning // In Proceedings of LREC. – 2006.
4. *Manning, C. D.* Stanford typed dependencies manual / C. D. Manning, M. C. De Marneffe // Technical report, Stanford University [Electronic resource]. – Stanford, California, 2008. – Mode of access : https://nlp.stanford.edu/software/dependencies manual.pdf. – Date of access : 01.03.2020.
5. *Das, B.* A Novel System for Generating Simple Sentences from Complex and Compound Sentences / B. Das, M. Majumder, S. Phadikar // International Journal of Modern Education and Computer Science. – 2018. – Vol. 10, № 1. – P. 57–64.

This paper describes the relevance of the problem of basic linguistic analysis of text information and the linguistic processor as the main component of its effective solution. The relevance of building a linguistic processor based on tools that are freely available is also outlined. A description of such tools and an algorithmic diagram demonstrating the solution for basic linguistic text analysis problem using the mentioned type of linguistic processor is given.