ИСПОЛЬЗОВАНИЕ ЕСТЕСТВЕННОГО ЯЗЫКА ПРИ СОЗДАНИИ ИМИТАЦИОННЫХ МОДЕЛЕЙ

С. В. Рудометов¹, В. В. Окольнишников²

¹ *Институт вычислительной математики и математической геофизики СО РАН, 630090, Новосибирск*

² *Институт вычислительных технологий СО РАН, 630090, Новосибирск*

УДК 519.685.1

DOI: 10.24411/9999-016A-2019-10066

В работе воспроизводятся действия человека, который пытается анализировать полностью неизвестный текст, в котором ему известно, какие слова и словесные конструкции обозначают сущности (предметы, существа), какие — действия, а какие — те или иные свойства сущностей или действий. При таком анализе, мысленно, человек начинает строить схему, в которой сначала «создает» сущности, затем «снабжает» их свойствами, указанными в тексте, и затем — выясняет их взаимодействие и его свойства.

Специалист в предметной области, разрабатывающий имитационную модель в среде имитационного моделирования MTSS [1], действует по похожей схеме. Сначала он принимает решение об использовании некоторых элементарных моделей (ЭМ), затем создает и размещает (визуально) некоторое количество экземпляров ЭМ (ЭЭМ), затем изменяет их свойства, затем графически соединяет их друг с другом. Использование естественного языка выглядит в таком случае как применение скриптового языка. Но есть и важное отличие: человеку не нужно учить этот новый язык. Он будет использовать хорошо известные ему и понятные языковые конструкции.

Ключевые слова: имитационное моделирование, естественный язык.

Введение

В последнее время активно развивается использование технологий искусственного интеллекта, таких как семантический анализ текстов на естественных языках.

Как мы уже отмечали ранее [2], «семантический анализ текста на естественном языке имеет смысл только в контексте какой-то задачи, сам по себе он не может существовать». Там же приводились наиболее известные задачи семантического анализа: машинный перевод, поиск смысловой информации в тексте (semantic web, определение эмоционального окраса текста, реферирование информации и т.д.), управление роботами или технологическим оборудованием.

Для предметного контекста можно использовать сущности, уже существующие в системах имитационного моделирования. Имеются в виду прежде всего субъекты (как они определены в DEVS-формализме) и их свойства.

Любое достаточно подробное и осмысленное описание проблемы, задачи и т.д. может быть использовано для выделения подмножеств *понятий*, их *свойств* и их *активностей*. Эти три связанных подмножества затем могут быть использованы например для генерации исходного кода элементарных моделей (ЭМ) для создания новой библиотеки системы MTSS.

Сгенерированные таким образом ЭМ скорее всего будут неполными. Для их финального наполнения, отладки и верификации все равно потребуется участие программиста — специалиста в имитационном моделировании.

1 DEVS-формализм и элементы системы MTSS

В основу системы имитационного моделирования MTSS положен так называемый DEVS-формализм ([3]). Он является развитием дискретно-событийного подхода. DEVS описывает поведение *системы* на двух уровнях. В своей основе (первый уровень) система в нотации DEVS-формализма является дискретно-событийной имитационной моделью. Система входных и выходных портов, а также способ соединения простых *субъектов* для образования имитационной модели является DEVS-формализмом (второй уровень).

Более формально, имитационная модель представляется в виде наборов

$$M = \{s_j^i, i \in Class, j \in Inst^i\}, \{o^k, k \in Object\} \quad (1)$$

экземпляров субъектов разных классов. Здесь *Class* — множество классов объектов, *Instⁱ* — множество экземпляров объектов класса *i*. Другими словами, множество *M* является набором экземпляров субъектов разных классов. Верхний индекс — номер класса, нижний индекс — номер экземпляра. Множество *Object* — это объекты, то есть сущности в имитационной модели, которые не участвуют в ее работе но с которыми могут взаимодействовать субъекты — например стенки, различные ограничители, разного рода указатели. Все эти классы определяются на основе декомпозиции имитируемой системы. *s_jⁱ* могут соединяться друг с другом и взаимодействовать с *o^k*.

Каждый субъект в нотации DEVS представляется в виде набора своих важных свойств:

$$s_j^i = \{S, ta, \delta_{int}, X, \delta_{ext}, Y, \lambda\} \quad (2)$$

Здесь *S* — фиксированный набор состояний (как определено в Discrete Event Simulation, или сокращенно DES), *ta* (или time advance) — время следующего исполнения субъекта, *δ_{int}* — набор входных портов, *X* — функция переключения состояний, как это определено для DES, *δ_{ext}* — набор выходных портов, *Y* — функция реакции на события во входных портах, *λ* — функция отображения внутреннего состояния из множества *S* в набор выходных сигналов.

2 Системы распознавания текста

Наиболее очевидным и как следствие — популярным способом синтаксического анализа фраз на естественном языке является т.н. «грамматика составляющих». Эта грамматика пытается разбить предложение на более мелкие структуры, затем их — на еще более мелкие структуры, пока не дойдет до отдельных слов или минимальных синтаксических структур. Каждая такая структура отмечается своим признаком, обозначающим ее роль в предложении (POS-tagging). Из такого определения следует, что результат синтаксического разбора исходного предложения в частности зависит от порядка слов в предложении. Этот подход успешно применяется для языков, в которых правилами жестко задан порядок слов (например — английский язык).

Другой возможный подход — грамматика зависимостей. В отличие от грамматики составляющих, результатом синтаксического разбора предложения в данной грамматике будет набор зависимостей между частями предложения (в частности, это подразумевает, что для предложения уже выполнен синтаксический разбор с использованием грамматики составляющих). Грамматика зависимостей выделяет в предложении значимые части (субъект, объект), и зависимости между ними. В данном случае порядок слов не так важен, например, как в русском языке.

И наконец третий, наиболее сложный подход — это системы на основе машинного обучения.

Сегодня системы распознавания текстов доступны как в виде библиотек, на и в виде онлайн-сервисов.

Примеры библиотек — Stanford NLP [4], Apache OpenNLP [5]. Библиотека должна быть включена в состав проекта в виде файлов библиотеки и файлов ресурсов. Существенной частью библиотеки является реализация языковой модели для какого-то естественного языка. Положительным моментом является то, что например обе эти библиотеки разрабатываются большим количеством инициативных групп, которые представляют разные естественные языки. И поэтому теоретически можно найти языковую библиотеку для своего языка. Но на практике получается следующая картина:

- хорошая языковая модель, как правило, очень большая. Она требует существенных ресурсов для своей работы — как времени загрузки, так и количества памяти. И если с временем загрузки еще можно (с трудом) смириться, то большой расход памяти в системе имитационного моделирования, которая сама по себе требует существенного количества памяти, является очень существенным ограничением.

- языковая модель оказывается «запаянной» в текущей версии MTSS. А значит для изменения или улучшения языковой модели требуется выпускать новые версии.

Другой вариант — использовать онлайн-сервисы, например Google NLP [6]. Такие сервисы абсолютно лишены недостатков библиотечных реализаций (загрузка, расход памяти), но требуют постоянного интернет-соединения. Интерфейс этих сервисов достаточно прост — он выглядит как единственный метод «распознай текст», с посылкой запроса и получением ответа по технологии Representational State Transfer (REST). Также стоит отметить, что такие сервисы используют наиболее развитые языковые модели. Детали их реализации не доступны для конечного пользователя, но для системы имитационного моделирования существующей языковой модели вполне достаточно. Из недостатков онлайн — сервисов можно еще отметить, что они, как правило, являются платными, либо имеют триальные ограничения (количество распознаваний в сутки, ограниченный набор языков).

Стоит отметить, что каждый из рассмотренных подходов (статистический или на основе машинного обучения) имеет несколько неожиданное ограничение. Их создатели обучали (или анализировали) очень правильные с языковой точки зрения тексты (Шекспир, Толстой, Некрасов...). Поэтому и тексты для распознавания тоже должны быть написаны безупречно с точки зрения исходного естественного языка. Как правило, технические тексты таковыми не являются, и поэтому для более точного распознавания будет обязательно требоваться коррекция входного текста на естественном языке. Если же язык спецификации не поддерживается, то потребуются идеально перевести ее на поддерживаемый язык (например английский).

3 Пример задачи для имитационного моделирования с использованием естественного языка

Система MTSS предполагает наличие специализированной библиотеки ЭМ для решения определенных прикладных задач. На данный момент написано уже порядка 10 таких библиотек, в областях складской логистики, угледобычи, исследования компьютерных сетей. Каждая из этих библиотек создавалась программистами-специалистами в имитационном моделировании для решения конкретного круга задач, в кооперации со специалистами в предметной области.

При создании библиотеки требуется написать достаточно много программного кода. Он достаточно однотипный, и поэтому можно попробовать автоматизировать его написание.

А именно, из описания новой библиотеки, сделанного на естественном языке, потребуется выделить (согласно формулам 1 и 2) набор *Class* классов элементарных моделей, и для каждого класса определить внутренние состояния *S*, функции переключения состояний (алгоритмы) *X* и *Y*.

При выделении понятий (NOUN), алгоритмов (VERB) и свойств требуется явно указать, чему будут соответствовать выделенные сущности. Другими словами, произвести *теггирование*.

Существует простой и эффективный способ выделения нужных понятий из размеченного текста на естественном языке, схожий с обычными регулярными выражениями. Реализации можно найти, например, в Stanford NLP, TokenSequencePattern [7] или библиотеке Natural Language Toolkit для языка Python [8]. После выделения набора сущностей мы сопоставляем найденные слова с набором строк, которым идентифицирует (именует) себя каждая ЭМ в конкретной библиотеке MTSS. Похожим образом выделяются затем свойства найденных ЭМ, и затем — их взаимодействия. Такой процесс требует нескольких (как минимум трех) итераций в анализе одного и того же текста.

Для предложения (из [2]) «The combine moves from an insert point at an angle A to the horizon up, and after achievement of a roof of layer — at an angle bedding of layer B — deep into layer» после анализа сервисом google nlp (далее просто анализатор) и после теггирования внутри приложения:

- субъект (NOUN): «Combine»
- объекты (NOUN, сущности с которыми взаимодействуют субъекты): непосредственно из фразы: «insert point», «angle», «horizon», «roof», «layer», «bedding of layer».
- алгоритмы (VERB): *moves* (<субъект>**combine**, <точка>**от** (insert point, speed, angle(A, horizon)), <точка>**до**(не определено), <направление> **направление**(up, angle(A, horizon))), *moves* (<субъект>**combine**, <точка>**от**(не определено), <точка>**до**(не определено), <направление> **направление**(forward, угол(B, layer))).

Здесь значения, заключенные в угловые скобки $\langle$ и $\rangle$, обозначают стереотипы данных и служат здесь для описания типов параметров.

4 Результат генерации

После теггирования полученные данные могут естественным образом быть применены для генерации кода ЭМ для новой библиотеки.

Алгоритмы функционирования после генерации будут представлены существующими, являющимися частью MTSS, опробованными реализациями модели движения с ускорением и торможением. Для моделей технологических процессов этого может оказаться достаточно. Но если это не так — данный код можно изменить.

Существенной проблемой остается визуальное представление элементов. При начальной генерации они будут представлены только в виде параллелепипедов, (форм, имеющих как положение в пространстве, так и линейные размеры). Если потребуется, в дальнейшем им можно придать любое графическое представление.

5 Дальнейшее развитие ЭМ

Как правило, такие сгенерированные ЭМ не удовлетворяют требованиям в постановке задач (хотя и являются вполне функциональными). Программист-специалист в имитационном моделировании может добавлять новые свойства ЭМ, реализовывать новые функциональные состояния ЭМ, изменять их графическое представление (2D и 3D). Результат генерации может рассматриваться в этом случае как достаточно полный «подстрочный перевод» того что хочет заказчик в программный код системы имитационного моделирования. А доработки ЭМ связаны с тем, что в работе специалиста в имитационном моделировании есть пункт «взаимодействие с заказчиком для выяснения очевидных для него но неочевидных для программиста деталей имитируемой системы».

6 Отличие от предыдущих работ

Работа [9] использовала фразы на естественном языке для конфигурирования конкретной имитационной модели. Работа [2] рассматривает аспекты использования естественного языка вообще, и предлагает использовать представление выделенных сущностей как простые, базовые элементы онтологии. Также она использует библиотеку Stanford NLP.

В данной работе используется сервис Google [6], и по сути своей результат является отдельным программным продуктом, не входящим в основную систему MTSS. Работа [2], может, и является более изящной в научном смысле, но с точки зрения практики лучше иметь именно такой генератор, как описан в данной статье. К сожалению, такой подход все-таки не позволяет полностью сократить «дистанцию программирования» (см. [2]), и является компромиссным, промежуточным между чистым использованием естественного языка и чистым программированием.

Список литературы

- [1] Рудометов С. В. Визуально-интерактивная система имитационного моделирования технологических систем. Вестник СибГУТИ. 2011. No 3. С. 14–27.
- [2] Рудометов С. В., Окольнішников В.В. Обобщенная онтология имитационного моделирования для создания имитационных моделей технологического процесса с использованием естественного языка. "Вычислительная и прикладная математика 2017"(ВПМ 2017). 25 — 30 июня 2017, Академгородок, Новосибирск, Россия. С. 744–749.
- [3] Bernard Zeigler (1987). "Hierarchical, modular discrete-event modelling in an object-oriented environment". Simulation. 49 (5): pp 219–230.
- [4] The Stanford NLP Group / [Электронный ресурс]. – Режим доступа: <https://nlp.stanford.edu/> (дата обращения: 29.04.2018).

-
- [5] Apache OpenNLP / [Электронный ресурс]. – Режим доступа: <https://opennlp.apache.org/> (дата обращения: 29.04.2018).
- [6] Cloud Natural Language / [Электронный ресурс]. – Режим доступа: <https://cloud.google.com/natural-language/> (дата обращения: 29.04.2018).
- [7] Stanford Log-linear Part-Of-Speech Tagger. URL: <https://nlp.stanford.edu/software/tagger.shtml>
- [8] E. K. Steven Bird, Edward Loper, Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit. O'Reilly Media, 2009, p. 504
- [9] Рудометов С. В., Окольнишников В.В. Использование онтологии технологических систем добычи угля для задания имитационных моделей на естественном языке. "Распределенные информационно-вычислительные ресурсы"(DICR-2014). 2–5 декабря 2014, Новосибирск, Россия.

*Рудометов Сергей Валерьевич — к.т.н., науч.сотр. Института
вычислительной математики и математической геофизики СО РАН;
e-mail: sergersw@gmail.com;*

*Окольнишников Виктор Васильевич — д.т.н., вед. науч.сотр.
Института вычислительных технологий СО РАН;
e-mail: okoln@mail.ru.*

Дата поступления — 31 мая 2019 г.