Разработка вопросно-ответной системы с использованием машинного обучения

Науменко Алексей Михайлович, студент;
Шелудько Сергей Дмитриевич, студент;
Юлдашев Роман Юльевич, студент;
Хлебников Николай Олегович, студент
Национальный исследовательский ядерный университет «МИФИ» (г. Москва)

Обоснована задача создания автоматизированной вопросно-ответной системы. Рассмотрены возможные подходы к решению задачи: метод векторного представления слов и метод синтаксических деревьев. Исследованы технологии word2vec, NLTK, pymorphy2, использованные при реализации системы. Описаны алгоритмы обучения лингвистических нейросетей: Continuous Bag of Words и Skip-gram.

Ключевые слова: *машинное обучение, вопросно-ответная система, синтаксическое дерево, word2vec, nltk, pymorphy2, python*

В ходе выполнения профессиональных и повседневных задач люди часто сталкиваются с необходимостью семантического сжатия по теме [1] больших объемов текстовой информации, то есть, выражаясь бытовым языком, поиска в тексте ответа на заданный вопрос. Задача полнотекстового поиска частично решена современными поисковым системами, но результаты такого поиска часто бывают неудовлетворительными. Причинами этого могут являться использование пользователем речевых конструкций естественного языка (например, использование вопросительных слов), высокая частота встречаемости искомых ключевых слов в просматриваемом тексте и другие проблемы, связанные с отсутствием семантического анализа вопроса и массива просматриваемых данных.

Неавтоматизированные подходы: преимущества и недостатки

На настоящий момент существуют подходы, позволяющие решать описанную задачу на конечных массивах данных и достаточной для выявления закономерностей подборке запросов к ним. В неавтоматизированном виде это может быть реализовано набором часто задаваемых вопросов (FAQ) по какой-либо теме: перед просмотром всего массива данных пользователь может обратиться к краткой выдержке из информации, к которой часто обращались другие пользователи. Данный подход требует от пользователя времени на прочтение перечня вопросов и выявление среди них аналогичного своему, не гарантируя нахождения такового. Больший уровень автоматизации обеспечен, например, в системах телефонной поддержки клиентов банков или операторов сотовой связи: пользователь выбирает тему своего вопроса из списка предложенных тем и соединяется с оператором поддержки, имеющим инструкцию в виде древовидной структуры для решения часто возникающих проблем. Оператор, задавая пользователю вопросы и анализируя его ответы, продвигается по инструкции и сообщает пользователю сведения, полученные в конечном листе инструкции. Такая система эффективно отвечает на вопросы пользователей, но её организация и поддержка предполагают большие затраты на персонал. В данной статье рассматриваются возможные способы автоматизированного решения задачи поиска ответа на вопросы пользователей по конечному массиву знаний, а также приводится механизм реализации вопросно-ответной системы с использованием машинного обучения.

Автоматизированные подходы: возможные способы реализации

В основе одного из возможных подходов лежит алгоритм поиска ответа в объеме текстовой информации, при этом системе необходимо гарантировать наличие однозначного ответа в непротиворечивом источнике информации. На настоящий момент полноценный семантический анализ с использованием только компьютерных средств невозможен, поскольку это является AI-полной задачей [2], предполагающей разработку искусственного интеллекта, сопоставимого по возможностям с человеческим. В данной статье рассмотрена реализация практики «frequently asked questions» в расширенном виде, то есть в структуру системы входит база вопросов по заданной тематике, которые могут задать пользователи. Каждому из них ставится в соответствие ответ, причем одному ответу могут соответствовать несколько вопросов. Таким образом, описанный подход сводит задачу нахождения ответа к поиску вопроса из базы, семантически близкого к заданному. Для того чтобы решить эту задачу необхо-

димо построить модели заданного вопроса и каждого вопроса из базы. Моделью может служить, например, синтаксическое дерево или геометрический вектор. В данном случае сравнение моделей является объективным показателем семантической близости вопросов.

Синтаксическое дерево — это построенный по определенному алгоритму граф, узлами которого являются отдельные части предложения, а связи между узлами обозначают их синтаксическую связь. В узле дерева могут располагаться таксономические единицы, отдельные слова предложения, или функциональные единицы, сочетания слов, которые при расщеплении перестают выполнять синтаксическую функцию. Существуют четыре основных алгоритма расположения узлов в графе: теория членов предложения, грамматика Теньера, грамматика зависимостей, грамматика непосредственно составляющих [3]. Теория членов предложения — это алгоритм, в котором в качестве вершины дерева выступает член предложения, не являющийся подчиненным по отношению ни к одной другой синтаксической единице. В соответствии с грамматикой Теньера вершиной синтаксического дерева является глагол-сказуемое, также вводятся понятия актанты — функциональной единицы обязательной по отношению к сказуемому,— и сирконстанты — необязательной (факультативной) функциональной единицы. Грамматика зависимостей — алгоритм, в котором в узлах дерева располагаются таксономические единицы, вершиной дерева является глагол-сказуемое или его аналитическая часть в случае составного глагола, все связи в дереве подчинительные. Грамматика непосредственно составляющих — алгоритм, в ходе выполнения которого каждая грамматическая единица делится на две более простых единицы; такое деление происходит вплоть до выделения в качестве узла отдельного слова, каждому узлу соответствует грамматический класс, среди которых все части речи, а также именная и глагольная составляющие.

Представление слова в векторном виде — сопоставление слова из словаря геометрическому вектору в пространстве R^n, где под словарём понимается пространство конечной размерности N, равной количеству всех представляемых в векторном виде слов. Задачей определения семантической близости между словами занимается дистрибутивная семантика. Увеличение размерности словесного векторного пространства способствует повышению точности определения смысловой близости, однако существует некоторая критическая размерность, превышая которую модель не приносит заметного увеличения точности. Обычно размерность вектора устанавливается в диапазоне от 100 до 1000. Любой алгоритм построения векторного пространства стремится к максимизации косинусного сходства между векторами семантически близких слов. Косинусное сходство определяется формулой:

$$similarity = cos(\theta) = \frac{A \cdot B}{AB} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

где A и B — вектора, расстояние между которыми вычисляется, θ — угол между ними. Примером семантического анализатора является word2vec, программное средство для построения словесных векторных пространств, разработанное компанией Google в 2013 году [4].

Word2vec как семантический анализатор

Word2vec основан на двухслойной нейронной сети прямого распространения, поэтому у пользователей существует возможность обучить сеть на собственных текстовых корпусах и, таким образом, получить наиболее подходящую для решения текущей задачи векторную модель. Результаты обучения модели зависят от выбранной пользователем модельной архитектуры. Всего в word2vec реализованы два алгоритма обучения: Continuous Bag of Words (CBOW) и Skip-gram.

При использовании архитектуры CBOW алгоритм предсказывает слово, исходя из его контекста, т.е. анализируя наборы слов, находящиеся по правую и левую стороны от данного. При этом результат работы алгоритма не зависит от порядка контекстных слов. Входным элементом в нейронную сеть выступает набор контекстных векторов w(t-k),.., w(t-1), w(t+1),..., w(t+k), а выходным вектором — w(t), где w(t) — вектор предсказанного на основе контекста слова. Архитектура Skip-gram отличается от CBOW тем, что предсказывает набор слов вокруг, основываясь на данном слове. Входным вектором выступает w(t), а выходным элементом — множество M = {w(t-k),.., w(t-1), w(t+1),..., w(t+k)}, где M — множество векторов. Каждое слово, соответствующее векторам из множества M, характеризует слово, соответствующее входному вектору. Схема работы алгоритмов CBOW и Skip-gram показана на рисунке 1.

Работу word2vec можно разделить на пять этапов. На первом этапе происходит статистическая обработка входного текстового корпуса, то есть для каждого слова рассчитывается количество вхождений его в исходный корпус. На втором этапе происходит сортировка слов по частоте вхождения, а также, в целях оптимизации работы с памятью, удаляются так называемые гапаксы — слова, встречающиеся редко в сравнении с другими словами текста,— результаты работы этапа сохраняются в хеш-таблице. На третьем этапе для сжатия данных к полученной хеш-таблице применяется код Хаффмана — алгоритм оптимального префиксного кодирования,— в результате чего чаще встречающиеся слова кодируются меньшим количеством бит, а менее часто — большим. На четвертом этапе происходит суб-сэмплирование самодостаточной выборки из текстового корпуса (например, предложения или абзаца), в ходе которого из выборки удаляются наиболее часто встречающиеся слова, поскольку такие слова обычно не несут значимого смысла. Операция суб-сэмплирования применяется для уменьшения времени обучения модели. На пятом этапе к получившейся выборке применяется один из алгоритмов обучения, рассмотренных выше: CBOW или Skip-gram.

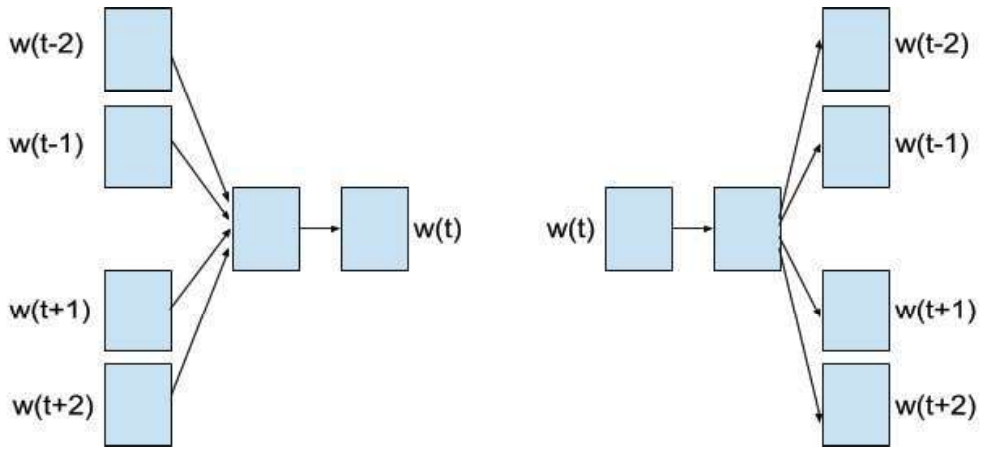

Рис. 1. **Схема работы алгоритмов CBOW и Skip-gram**

Процесс реализации вопросно-ответной системы

В структуру системы входят следующие элементы: управляющие скрипты; обученная на Национальном корпусе русского языка векторная Skip-gram-модель в 300-мерном пространстве [5,6], в которую входит более 184000 лемм; база ответов на вопросы, сами вопросы, а также их векторное представление.

Технология Word2vec применима как к отдельным словам, так и к текстам, в данном случае являющимся вопросительными предложениями. Из этого ясно, что на основе любой текстовой выборки можно построить её векторное представление путём суммирования векторов слов, входящих в выборку. Каждое слово из предложения должно быть лемматизировано, так как в векторную модель входят только леммы, то есть слова в словарной форме, и так как форма не имеет значения в определении смысла слова. Некоторые части речи, как существительные, глаголы и прилагательные, в большинстве случаев являются семантически значимыми, тогда как другие: предлоги, союзы и местоимения,— не несут смысловой нагрузки, поэтому при анализе предложения к ним применяется фильтрация, в ходе которой из него исключаются так называемые стоп-слова [7] и знаки препинания. Пример подготовки предложения к построению векторной модели изображен на рисунке 2.

Рис 2. **Пример анализа предложения**

Управляющие скрипты написаны на языке Python. Лемматизация производится средствами библиотеки pymorphy2 [8], которая в настоящий момент способна обрабатывать до 100000 слов в секунду, при этом потребление оперативной памяти колеблется от 10 до 20 Мб [9]. Набор используемых стоп-слов взят из библиотеки Natural Language Toolkit [10], которая используется в проектах, связанных с компьютерной лингвистикой и машинным обучением, и предназначена для обработки естественного языка.

Процесс реализации вопросно-ответной системы можно условно разделить на несколько этапов. Так как система базируется на практике «frequently asked questions», на первом этапе осуществляется проектирование базы вопросов и ответов. На втором этапе в базу заносится векторное представление каждого вопроса, построенное по алгоритму, описанному ранее. Затем этот же алгоритм применяется к вопросу, заданному пользователем системы. Таким образом, система получает необходимую для определения семантической близости информацию, на основе которой может делать предположения о смысловом сходстве или различии заданного вопроса и вопросов из базы. На заключительном этапе для каждого вопроса из базы и заданного пользователем вопроса рассчитывается косинусное сходство. Результатом работы системы будет ответ на тот вопрос из базы, косинусное сходство с которым максимально. Схема принципа работы системы изображена на рисунке 3.

Заключение

Системы семантического анализа способны найти применение в большом количестве социальных, научных

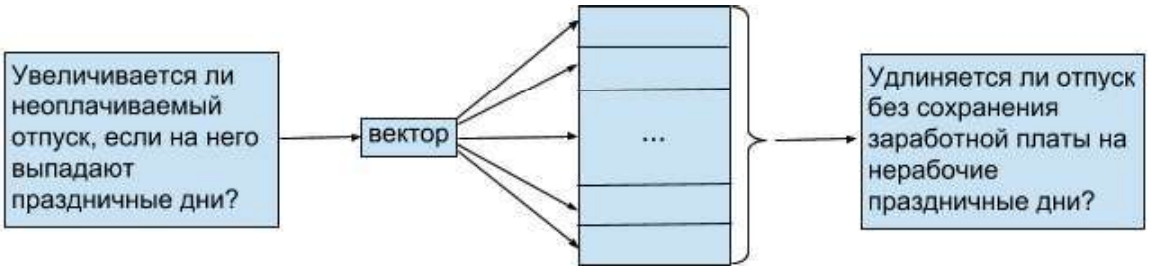

Рис 3. **Пример работы системы**

и бизнес-задач. Представленный подход к их реализации, основанный на технологиях word2vec, nltk и pymorphy2, имеет преимущества в сравнении с широко использующимися на настоящий момент решениями. Word2vec и nltk распространяются под свободной лицензией Apache 2.0, pymorphy2 распространяется под свободной лицензией MIT, что позволяет беспрепятственно использовать их при разработке коммерческих продуктов. Таким образом, построение вопросно-ответных систем на основе нейросетей является перспективным направлением в практическом применении механизмов машинного обучения.

Литература:

1. Ceglarek, D.: Semantic Compression for Text Document Processing. In: Nguyen, N.T. (ed.) Transactions on Computational Collective Intelligence XIV, pp. 20–48. Springer, Heidelberg (2014).
2. Eric S. Raymond. The New Hacker's Dictionary. — MIT Press, 1996. — P. 38–39. — 547 p
3. Касевич В. Б. Структура предложения // Элементы общей лингвистики. — М.: Наука, 1977.
4. Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff (2013). Distributed representations of words and phrases and their compositionality.
5. Kutuzov, Andrey and Andreev, Igor. «Texts in, meaning out: neural language models in semantic similarity task for Russian», in Proceedings of the Dialog 2015 Conference. Moscow, Russia (2015).
6. http://ling.go.mail.ru — официальный сайт проекта RusVectōrēs (дата последнего обращения 22.02.2017).
7. Manning, Christopher; Raghavan, Prabhakar; Schütze, Hinrich: An Introduction to Information Retrieval, p. 27. Cambridge University Press, Cambridge, England (2009).
8. Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp 320–332 (2015).
9. http://pymorphy2.readthedocs.io — официальный сайт проекта pymorphy2 (дата последнего обращения 22.02.2017).
10. http://www.nltk.org/ — официальный сайт проекта Natural Language Toolkit (дата последнего обращения 22.02.2017).