Подготовка набора данных для вопросно-ответного поиска по базе знаний. Первый этап: сопоставление сущностей

В.О. Кораблинов [1,2]

[1] Университет ИТМО, [2] JetBrains Research

vladislav.korablinov@gmail.com

Аннотация

На текущий момент вопросно-ответный поиск по базам знаний является активно развивающейся областью. Новые подходы демонстрируют стабильное повышение качества, однако такое развитие было бы невозможно без разработки наборов данных, позволяющих обучать модели, измерять их качество и ставить все более сложные задачи. К сожалению, все существующие наборы данных содержат вопросы только на английском языке, что ограничивает исследования в этой области для других языков. Мы хотим заполнить этот пробел, разработав набор данных для оценки методов вопросно-ответного поиска по базам знаний на русском языке.

В данной работе описывается способ создания такого набора данных с помощью краудсорсинга, одним из ключевых этапов которого является выделение в текстах вопросов и ответов упоминаний сущностей и их сопоставление с сущностями базы знаний. Разработанный нами алгоритм позволяет строить списки таких возможных упоминаний и находить правильную сущность в 95% случаев. При этом алгоритм автоматически связывает фрагменты текста с сущностями базы знаний Wikidata. Полученные списки в дальнейшем будут использованы для получения разметки вопросов и ответов, необходимой для создания нового набора данных.

Ключевые слова: вопросно-ответный поиск, базы знаний, графы знаний, сопоставление сущностей, краудсорсинг

Библиографическая ссылка: Кораблинов В.О. Подготовка набора данных для вопросно-ответного поиска по базе знаний. Первый этап: сопоставление сущностей // Компьютерная лингвистика и вычислительные онтологии. Выпуск 4 (Труды XXIII Международной объединенной научной конференции «Интернет и современное общество», IMS-2020, Санкт-Петербург, 17 – 20 июня 2020 г. Сборник научных статей). — СПб.: Университет ИТМО, 2020. С. 98-108. DOI: 10.17586/0000-0000-2020-4-98-108

Введение

Задача вопросно-ответного поиска – предоставить точный и исчерпывающий ответ на вопрос, заданный человеком на естественном языке. В этой задаче выделяется два основных направления – общий вопросно-ответный поиск и вопросно-ответный поиск по базам знаний. В первом поиск производится по большой коллекции текстовых документов и разделен на две стадии – 1) извлечение потенциально релевантных параграфов; 2) выделение в параграфе фрагмента текста, содержащего ответ. Значительным отличием вопросно-ответного поиска по базам знаний, как и следует из названия, является использование специализированных баз знаний в качестве источников информации. База знаний представляет собой множество фактов, которые обычно структурированы в виде троек «субъект-предикат-объект» или «субъект-предикат-литерал», например (Москва, является_столицей, Россия) и (Александр_Сергеевич_Пушкин, дата_рождения, 06-06-

1799). Субъекты и объекты принято называть сущностями, а предикаты - отношениями. Любая сущность и отношения имеют свой собственный уникальный идентификатор, а также одно или несколько названий на естественном языке. Основной задачей вопросно-ответного поиска по базам знаний является преобразование вопроса на естественном языке в конструкцию некоторого формального языка, например, SPARQL-запрос или терм лямбда-исчисления. Среди популярных применений вопросно-ответного поиска – быстрые ответы поисковых машин. Также известным примером является программа IBM Watson [1], обыгравшая сильнейших игроков в викторину «Jeopardy!».

Для достижения значительного прогресса в любых задачах, связанных с естественным языком, в том числе, вопросно-ответного поиска, очень важны наборы данных с высококачественной разметкой. С момента появления набора данных для общего вопросно-ответного поиска Stanford Question Answering Dataset (SQuAD) [2] было выпущено достаточно много других различных наборов данных для решения и оценки этой задачи. Для вопросно-ответного поиска по базам знаний доступно гораздо меньше наборов данных, при этом за последние несколько лет было разработано множество различных методов [3] качество работы которых необходимо оценивать. Недавние статьи [4] упоминают 10 наборов данных, из которых лишь 5 состоят из более чем 5,000 вопросов. При этом все вопросы в этих наборах сформулированы на английском языке, что не позволяет эффективно обучать и оценивать модели для других языков. В то же время авторы недавнего исследования [5] выделяют многоязычный вопросно-ответный поиск по базам знаний в качестве одного из главных направлений развития области. В связи с этим перед нами возникает задача создания наборов данных на русском языке.

Для создания такого набора требуется множество пар «вопрос-ответ», которые можно получить, например, из коллекций викторин. Также необходимо сделать выбор самой базы знаний, по которой будет производиться поиск ответов. Наиболее популярными являются три базы знаний – Freebase [6], DBPedia [7] и Wikidata [8]. Несмотря на то, что большинство существующих датасетов построено над Freebase, с 2015 года Google прекратил поддержку этой базы знаний, поэтому информация в ней уже сейчас является во многом устаревшей. Большая часть данных Freebase была перенесена в базу знаний Wikidata. Wikidata поддерживается крупным сообществом Wikimedia, позволяет пользователям свободно редактировать информацию и на данный момент является наиболее активно развивающейся. База знаний DBPedia редактируется только небольшим числом поддерживающих ее экспертов, из-за чего сильно проигрывает Wikidata в объеме имеющихся данных. Поэтому мы сделали выбор в пользу Wikidata в качестве основной базы знаний.

Однако наиболее ценной частью набора данных является разметка каждой такой пары. В простейшем случае разметка состоит из запроса на языке SPARQL и идентификаторов сущностей в базе знаний, являющихся ответом. Для создания разметки по вопросу вручную требуется знание структуры базы знаний и языка SPARQL, кроме того, ручная разметка займет у экспертов очень много времени. Одним из популярных способов ускорить процесс разметки является ее декомпозиция на последовательность более простых шагов, каждый из которых может быть легко выполнен любым человеком после небольшой (5-10 минут) подготовки. Эти шаги могут быть сформулированы в виде заданий на краудсорсинговых платформах и будут выполнены быстро и относительно дешево.

Один из способов автоматического построения SPARQL-запросов, предложенный авторами датасета FreebaseQA [9], также основанного на вопросах викторин, заключается в следующем. Нам необходимо понять, какие сущности упоминаются в тексте вопроса и тексте ответа, после чего выбрать правильное отношение между ними. Например, в вопросе «Кто снял фильм “Бриллиантовая рука”»? должны быть выделены сущности Q1999930 (“Бриллиантовая рука”) и Q11424 (фильм), его ответ – Леонид Гайдай – должен быть сопоставлен сущности Q312480, а затем между сущностями Q1999930 и Q312480

нужно найти отношение P57 (режиссер). В рамках данной работы мы рассмотрим задачу сопоставления сущностей. Решение этой задачи зачастую разбивается на два этапа: выделение сущностей в тексте и последующее нахождение соответствующей выделенному фрагменту сущности в базе знаний. Для первого этапа наиболее часто используются n-граммные модели [10; 11; 12; 13]. Еще одним популярным инструментом являются модели распознавания именованных сущностей, основанные на нейронных сетях [14; 15; 16; 17]. Для сопоставления фрагменту сущности практически все методы используют индекс названий сущностей, производя в нем поиск по строковой схожести. Предложенный в данной работе метод сочетает в себе ориентированные на полноту приемы с приемами, ориентированными на точность. В то же время автоматически может быть выделен лишь список сущностей, которые потенциально могут содержаться в вопросе или ответе, однако из-за языковых неоднозначностей часто нельзя без помощи человека точно определить, какие из них действительно упоминаются в тексте. Для решения этой проблемы мы используем задание на краудсорсинговой платформе.

Далее статья организована следующим образом. В разделе 2 описываются особенности подготовки заданий для краудсорсинговых платформ и объясняется влияние этих особенностей на решение задачи выделения сущностей. В разделе 3 описана схема разработанного нами алгоритма генерации списка сущностей-кандидатов, а также различные нюансы его реализации. Раздел 4 описывает полученные результаты.

1. Особенности разметки данных с помощью краудсорсинга

1.1. Краудсорсинговая платформа

Краудсорсинговая платформа представляет собой веб-сервис, позволяющий заказчикам создавать проекты и загружать в них задания, которые будут выполнены автоматически подобранными исполнителями. Перед выполнением заданий исполнители изучают инструкцию и проходят небольшое обучение, подготовленное заказчиком. Обычно проекты настраиваются таким образом, чтобы каждое задание было выполнено несколькими исполнителями, после чего их оценки агрегируются с помощью предоставляемых платформой алгоритмов. Для мотивации исполнителя и контроля качества выполнения заданий платформы предоставляют разнообразные инструменты, однако ключевым моментом для получения наиболее высокой точности ответов является простота и понятность самого задания. Наиболее простыми на практике оказываются задания на классификацию. Именно к такому заданию мы сведем нашу задачу выделения сущностей.

1.2. Выделение сущностей как задача классификации

Мы будем выделять сущности в вопросах и ответах в два этапа. На первом этапе с помощью разработанного нами алгоритма мы построим список сущностей-кандидатов. На втором этапе мы запустим задание по выбору правильных кандидатов на краудсорсинговой платформе.

Задачи выделения сущностей в текстах вопросов и текстах ответов отличаются тем, что в вопросе может упоминаться несколько сущностей, а в ответе – только одна. Поэтому для текстов ответов мы можем просто сгенерировать список кандидатов, а затем просить исполнителей выбрать из этого списка правильную сущность. При этом число вариантов должно быть относительно небольшим. Кроме того, чтобы облегчить задачу исполнителя, и тем самым уменьшить время, потраченное на разметку, мы должны постараться показывать варианты в порядке уменьшения вероятности их упоминания.

Для текстов вопросов мы могли бы сформулировать похожее задание, но просить исполнителей отмечать несколько кандидатов, каждый из которых упоминается в вопросе. Однако у такого подхода есть существенные недостатки.

Во-первых, исполнителю придется внимательно изучать различия между кандидатами, имеющими похожие названия, что существенно замедлит процесс разметки. Во-вторых, ответы на задания с множественными ответами не могут быть автоматически агрегированы, не позволяя настроить некоторые важные правила контроля качества. Поэтому в данном случае наиболее подходящим решением является сведение выбора кандидатов к бинарной классификации. Мы будем показывать каждому исполнителю вопрос и только одного из кандидатов, и просить определить, упоминается этот кандидат в тексте вопроса. При использовании этого подхода увеличивается количество заданий, однако каждое из них становится намного проще, что в результате приводит к более быстрому выполнению всего набора заданий. Тем не менее, количество кандидатов также необходимо ограничить некоторым небольшим числом.

2. Алгоритм генерации сущностей-кандидатов

2.1. Общая структура алгоритма

Наш алгоритм состоит из предварительного этапа подготовки данных и четырех основных этапов генерации списка кандидатов.

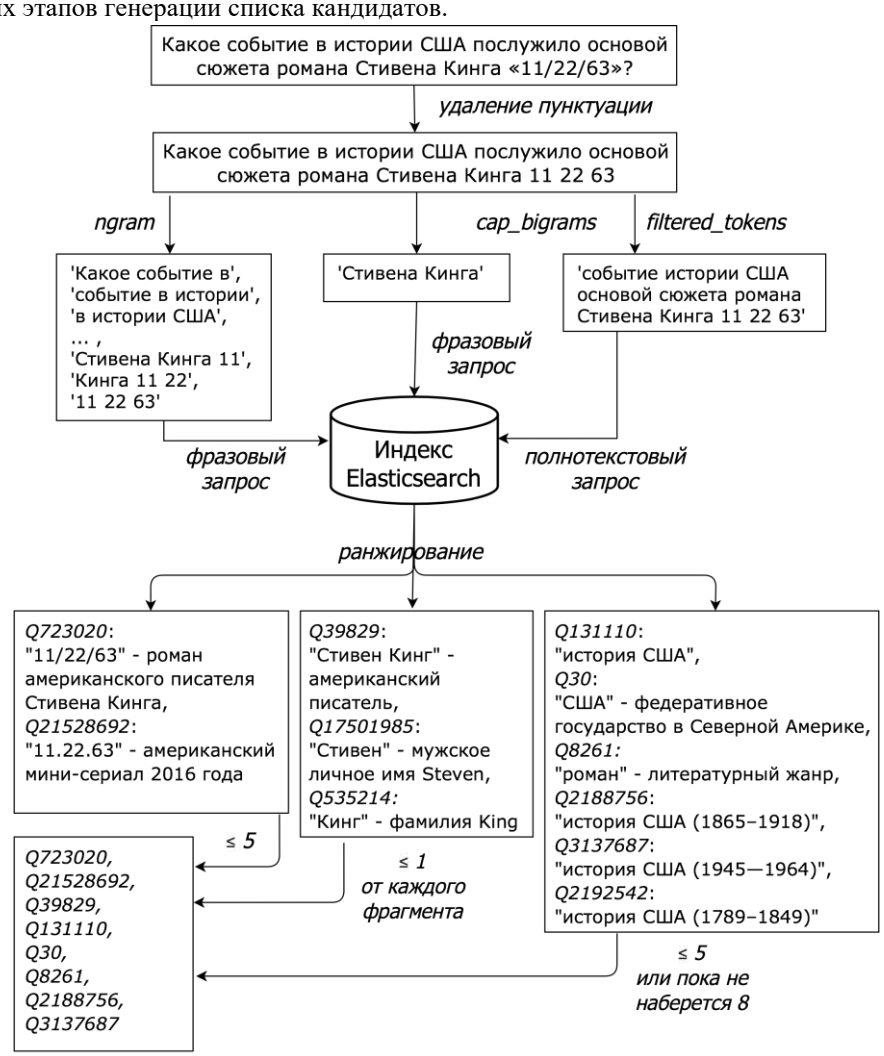

Рис. 1. Применение алгоритма генерации сущностей-кандидатов к тексту вопроса

На предварительном этапе собираются названия всех известных сущностей и индексируются поисковым движком. На первом этапе генерации кандидатов в тексте вопроса или ответа выделяются фрагменты, которые могут содержать упоминание сущности. На втором этапе эти фрагменты используются для формирования запросов к поисковому индексу. На третьем этапе результаты запросов проходят процедуру ранжирования. На заключительном, четвертом, этапе из ранжированных результатов нескольких запросов формируется окончательный список кандидатов. Общая схема алгоритма показана на рис. 1.

2.2. Подготовка поискового индекса

Для решения задачи выделения сущностей необходимо иметь список всех имеющихся в базе знаний сущностей, из которого мы будем выбирать кандидатов для конкретного текста. В рамках задачи подготовки русскоязычного набора данных нас интересуют только те сущности Wikidata, для которых имеется русскоязычное название. Однако не все из них являются полезными - многие сущности выражают служебные понятия проектов Wikimedia и не несут фактологической информации о мире. Мы выделили 6 типов таких сущностей - страница значений Wikimedia, категория Wikimedia, статья Wikinews, список статей Wikimedia, словарная запись и энциклопедическая статья. Сущность в Wikidata может быть известна под несколькими именами, например, *Земля* также имеет названия *Голубая планета*, *Третья планета* и *мир*. С помощью серии SPARQL-запросов для всех подходящих сущностей был получен идентификатор и список названий. Всего мы получили 4,114,595 уникальных сущностей и 5,430,657 различных названий.

Далее необходимо было построить поисковый индекс на полученных данных. В качестве поискового движка был выбран Elasticsearch, предоставляющий широкий набор настроек для индексирования и выполнения запросов, а также имеющий удобный API для многих языков программирования. Мы представили наши сущности как набор текстовых документов. Каждое полученное название сущности выступало в качестве отдельного документа, а в качестве метаинформации документа был прикреплен соответствующий идентификатор сущности. Перед индексированием также принято производить морфологическую обработку документов, причем обработка документов и последующих запросов должна производиться одним и тем же методом. Существующие морфологические анализаторы для русского языка работают достаточно медленно – обработка одного запроса занимает несколько секунд, а запросов в дальнейшем мы будем делать много, поэтому было принято решение воспользоваться встроенным в Elasticsearch простым анализатором Snowball [18] для русского языка.

2.3. Фрагментация текста

Итак, мы переходим к непосредственному описанию алгоритма генерации кандидатов. На первом этапе мы хотим выделить в тексте фрагменты, которые могут содержать упоминание сущности. Хорошо известна задача выделения именованных сущностей, которая лучше всего решается с помощью обучения нейронных сетей [19]. Подход с использованием таких моделей для фрагментации текста при сопоставлении сущностей применялся в нескольких работах, посвященных созданию наборов данных [9; 16; 20]. Основным недостатком этого подхода является неспособность выделять сущности, выражающие классы или названия предметов, например *мультипликационный фильм* или *молоток*. Также готовые модели плохо справляются с определением границ фрагментов в случаях, когда некоторая именованная сущность является частью правильной сущности. Например, в вопросе «*Кто написал музыку гимна России?»* будет выделен фрагмент *России*, в то время как искомым является фрагмент *гимна России*. Для устранения указанных недостатков и повышения полноты получаемых списков сущностей-кандидатов

мы используем такую модель лишь в качестве одного из способов фрагментации, комбинируя этот способ с другими.

Для удобства работы с текстом мы убирали из него все знаки препинания и разбивали по пробелам на токены. Получившийся список токенов в дальнейшем будем называть *tokens*. Из этого списка мы получали три различных типа фрагментов.

Фрагменты первого типа мы получали, извлекая из списка *tokens* все n-граммы. Такие фрагменты позволяют найти в индексе точное упоминание сущности. Для получения качественного результата необходимо выбрать подходящее число n. Для ответов хорошим значением является 2, так как большая часть текста ответа является упоминанием сущности. Тексты вопросов содержат более разнообразные языковые конструкции, поэтому многие 2-граммы не содержат упоминаний сущностей. Тем не менее, эксперименты показали, что уже при n, равном 3, мы получаем приемлемую точность. Список полученных таким образом фрагментов мы будем называть *ngrams*.

Фрагменты второго типа мы получали с помощью модели выделения именованных сущностей. Полученный список таких фрагментов мы будем называть *ner*.

Третий тип был представлен ровно одним фрагментом. Это единственный тип фрагмента, который не является наборов последовательных токенов из списка *tokens*. Этот фрагмент необходим для обнаружения сущностей, которые коротко выражаются (1-2 токена) и не являются именованными. Для получения такого фрагмента мы получили для каждого из токенов из списка tokens его часть речи с помощью инструмента MyStem, а затем удаляли токен, если его часть речи была одной из следующих: союз, междометие, частица, предлог, местоимение, наречие или глагол. Токены этих частей речи могут быть частью названия сущности, состоящего из нескольких, обычно более чем двух токенов, однако сами по себе крайне редко выражают какую-либо сущность. Получившийся фрагмент мы будем называть *filtered_tokens*.

2.4. Создание запросов к поисковому движку

На следующем этапе из полученных фрагментов мы формировали запросы к индексу Elasticsearch. Напомним, что каждый фрагмент представляет собой список токенов. Для создания текста запроса каждый список токенов конкатенировался в строку с пробелами между токенами, таким образом, мы получили строку из каждого фрагмента. Далее эти строки были использованы в качестве текстов запросов двух различных типов, предоставляемых поисковым движком.

Первый тип – обычные полнотекстовые запросы, позволяющие искать в индексе документы, содержащие один или несколько токенов запроса, и ранжирующие результаты по метрике BM25. Elasticsearch позволяет также настроить для этого типа запроса так называемый нечеткий поиск. При его использовании токены, незначительно отличающиеся друг от друга, будут считаться идентичными, что позволяет справляться с опечатками в исходном тексте и ошибками встроенного морфологического анализатора. Отличия между токенами оцениваются с помощью расстояния редактирования, при этом разрешенное значение метрики является динамическим и зависит от длины токенов. Чтобы избежать случайных совпадений, вызванных тем, что в числе операций для подсчета расстояния редактирования есть операция перестановки соседних букв, мы также произвели настройку, запрещающую токенам считаться одинаковыми, если различаются их первые буквы. Настроенный таким образом полнотекстовый поиск был применен к фрагменту *filtered_tokens*, так как он может содержать несколько различных упоминаний сущностей в неопределенном порядке.

Второй тип – фразовые запросы, позволяющие найти в индексе точное вхождение текста запроса в документ. Этот тип запросов использовался для фрагментов *ngram* и *ner*, так как эти фрагменты должны соответствовать конкретным упоминаниям сущностей. Все строки, полученные из фрагментов *ngram*, были включены в один запрос, в то время

как каждая из строк, полученных из фрагментов *ner*, порождала свой собственный фразовый запрос.

2.5. Ранжирование результатов запросов

На следующем этапе результаты запросов должны быть ранжированы таким образом, чтобы выше оказались сущности, более вероятно упоминающиеся в тексте. Для ранжирования мы использовали два параметра. Первый – оценка BM25, выданная Elasticsearch. Она показывает, насколько текст запроса соответствовал тексту документа (то есть названию некоторой сущности). Второй – количество месячных просмотров статьи Википедии, соответствующей сущности. Так мы узнаем, насколько популярной является это сущность.

Изначально результаты запроса упорядочены по оценке BM25 по убыванию. Для ранжирования с использованием статистики просмотров Википедии мы ввели численный параметр *rate* и разбили все результаты запроса на группы, внутри которых оценки результатов отличаются не более чем в *rate* раз. После этого внутри каждой группы мы переупорядочили результаты по убыванию количества просмотров соответствующей статьи Википедии.

2.6. Формирование списка сущностей-кандидатов

На заключительном этапе мы собирали окончательный список сущностей-кандидатов из ранжированных результатов запросов. Для управления размером этого списка и вкладом, вносимым каждым из типов запросов, мы ввели три параметра – *p, f* и *m*, смысл которых мы укажем далее.

Сначала в список были добавлены первые *p* сущностей из запроса, сформированного из фрагментов *nrgam*. Затем из каждого из запросов, сформированных их фрагментов *ner,* было взято по одной первой сущности, сущности были отсортированы по числу просмотров статьи Википедии и получившийся список был добавлен к результирующему. Наконец, в список были добавлены первые *f* сущностей из полнотекстового запроса. После этого были убраны все дубликаты сущностей в результирующем списке.

Таблица 1. Значения параметров для формирования списка кандидатов

Тип текста	p	f	m
Текст вопроса	5	5	8
Текст ответа	4	3	5

Если размер списка оказался меньше, чем m, мы дополняли его следующими сущностями из полнотекстового запроса до величины m, либо пока не исчерпаются результаты запроса. Значения параметров для разных типов текстов приведены в табл. 1.

3. Оценка результатов работы алгоритма

Для того, что оценить качество работы разработанного алгоритма, мы с его помощью сгенерировали списки сущностей-кандидатов для 14,435 пар “вопрос-ответ”. Средняя длина списка кандидатов для вопросов составила 9.37, средняя длина списка кандидатов для ответов – 5.4. Следуя разделу 2.2, мы создали задания на краудсорсинговой платформе Яндекс.Толока и получили результаты их выполнения.

Так как для ответов правильной всегда должна являться только одна сущность, мы изучили, как высоко оказывался правильный ответ в нашем списке кандидатов. Результаты приведены в табл. 2.

Таблица 2. Распределение позиций правильной сущности в списке кандидатов

Позиция	Количество
1	7328
2	1204
3	492
4	250
5	201
6	66
7	44
8	59
9	8
10	2
11	1
Не найдено	4780

Видим, что наши списки были действительно хорошо упорядочены, и наиболее вероятные кандидаты были расположены выше наименее вероятных. Чтобы выяснить, почему для многих ответов мы не смогли найти правильную сущность, мы проанализировали 100 случайных ответов и их списки кандидатов. Оказалось, что только в 14 случаях причиной стала ошибка нашего алгоритма, например, для ответа *Бородин* было найдено 7 населенных пунктов с названием Бородино, а композитор Александр Порфирьевич в списке кандидатов отсутствовал.

Таблица 3. Причины отсутствия правильной сущности в списке кандидатов

Причина	Кол-во
В тексте ответа не содержится важная часть названия сущности (например, «Ивановская» вместо «Ивановская область»)	41
Текст ответа, соответствующий очень многим сущностям, при этом правильная сущность очень редка (например, «Камчатка» в значении названия судна)	21
Ошибки разметки	10
Отсутствие упомянутой сущности в Wikidata (например, «Лада» в значении автомобильной марки)	9
В тексте сущность упоминается кириллицей, а в Wikidata есть только написание латиницей (например, «Порше» и «Porsche»)	5

Распределение остальных причин показано в табл. 3.

Заключение

Разработанный нами алгоритм генерации списков сущностей-кандидатов хорошо проявил себя на практике, обеспечивая достаточную полноту при относительно небольшой средней длине списков. В дальнейшем, используя этот алгоритм, мы сможем автоматически построить SPARQL-запросы для вопросов, получив необходимую разметку, и тем самым сформировав новый набор данных для оценки методов вопросно-ответного поиска по базам знаний. Заметим также, что высокое качество алгоритма позволяет использовать его в качестве системы выделения сущностей в предложениях наподобие DBPedia Spotlight [21] – для этого достаточно отсекать кандидатов из списка по некоторому порогу полученной оценки.

Литература

[1] Ferrucci D. A. Introduction to “this is watson” // IBM Journal of Research and Development. 2012. Т. 56, №. 3.4. P. 1-15.

[2] Rajpurkar P. et al. Squad: 100,000+ questions for machine comprehension of text // arXiv preprint arXiv:1606.05250. 2016.

[3] Diefenbach D. et al. Core techniques of question answering systems over knowledge bases: a survey // Knowledge and Information systems. 2018. Т. 55, №. 3. P. 529-569.

[4] Dubey M. et al. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia // International Semantic Web Conference. Springer, Cham. 2019. P. 69-78.

[5] Chakraborty N. et al. Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs // arXiv preprint arXiv:1907.09361. 2019.

[6] Bollacker K. et al. Freebase: a collaboratively created graph database for structuring human knowledge // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 2008. P. 1247-1250.

[7] Auer S. et al. Dbpedia: A nucleus for a web of open data // The semantic web. Springer, Berlin, Heidelberg, 2007. P. 722-735.

[8] Vrandečić D., Krötzsch M. Wikidata: a free collaborative knowledgebase // Communications of the ACM. 2014. Т. 57, №. 10. P. 78-85.

[9] Jiang K., Wu D., Jiang H. FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style Question-Answer Pairs with Freebase // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019. P. 318-323.

[10] SZ H. et al. CASIA@ V2: A MLN-based question answering system over linked data. 2014.

[11] Ruseti S. et al. QAnswer-Enhanced Entity Matching for Question Answering over Linked Data // CLEF (Working Notes). 2015. P. 28-35.

[12] Beaumont R., Grau B., Ligozat A. L. SemGraphQA@ QALD5: LIMSI participation at QALD5@ CLEF. 2015.

[13] Shekarpour S. et al. Sina: Semantic interpretation of user queries for question answering on interlinked data // Journal of Web Semantics. 2015. Т. 30. P. 39-51.

[14] Cabrio E. et al. QAKiS: an open domain QA system based on relational patterns. 2012.

[15] Zhang Y. et al. Question answering over knowledge base with neural attention combining global knowledge information // arXiv preprint arXiv:1606.00979. 2016.

[16] Bao J. et al. Constraint-based question answering with knowledge graph // Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. 2016. P. 2503-2514

[17] Petrochuk M., Zettlemoyer L. Simplequestions nearly solved: A new upperbound and baseline approach // arXiv preprint arXiv:1804.08798. 2018.

[18] Porter M. F. Snowball: A language for stemming algorithms. 2001.

[19] Yadav V., Bethard S. A survey on recent advances in named entity recognition from deep learning models // arXiv preprint arXiv:1910.11470. 2019.

[20] Yih W. et al. The value of semantic parse labeling for knowledge base question answering // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2016. P. 201-206.

[21] Mendes P. N. et al. DBpedia spotlight: shedding light on the web of documents // Proceedings of the 7th international conference on semantic systems. 2011. P. 1-8.

Dataset Creation for Question Answering Over Knowledge Bases. First Stage: Entity Linking

V.O. Korablinov [1,2]

[1] ITMO University, [2] JetBrains Research

Question answering over knowledge bases has been an active area of research over the past years. Regularly emerging new approaches show a steady increase in quality, but this would not be possible without development of datasets that allow training models, evaluating them and setting new challenging tasks. Unfortunately, all existing datasets are available only in English, which limits research in this area to other languages. We want to close this gap by providing a dataset for the Russian language.
In this paper we describe an approach to create such dataset using crowdsourcing. One of the key stages of our approach is to identify mentions of entities in the texts of questions and answers and link them to the entities in knowledge base. Our algorithm allows us to build lists of such possible mentions. The algorithm also automatically links extracted text spans to Wikidata entity identifiers. Evaluation shows that we find proper linking in 95% of cases. The resulting lists will later be used to create questions and answers annotations needed for dataset creation.

Keywords: NLP, question answering, knowledge base, knowledge graph, entity linking, crowdsourcing

Reference for citation: Korablinov V.O. Dataset creation for question answering over knowledge bases. First stage: entity linking // Computer Linguistics and Computing Ontologies. Vol. 4 (Proceedings of the XXIII International Joint Scientific Conference «Internet and Modern Society», IMS-2020, St. Petersburg, June 17-20, 2020). - St. Petersburg: ITMO University, 2020. P. 98 – 108. DOI: 10.17586/0000-0000-2020-4-98-108

References

[1] Ferrucci D. A. Introduction to “this is watson” // IBM Journal of Research and Development. 2012. T. 56, №. 3.4. P. 1-15.
[2] Rajpurkar P. et al. Squad: 100,000+ questions for machine comprehension of text // arXiv preprint arXiv:1606.05250. 2016.
[3] Diefenbach D. et al. Core techniques of question answering systems over knowledge bases: a survey // Knowledge and Information systems. 2018. T. 55, №. 3. P. 529-569.
[4] Dubey M. et al. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia // International Semantic Web Conference. Springer, Cham. 2019. P. 69-78.
[5] Chakraborty N. et al. Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs // arXiv preprint arXiv:1907.09361. 2019.
[6] Bollacker K. et al. Freebase: a collaboratively created graph database for structuring human knowledge // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 2008. P. 1247-1250.
[7] Auer S. et al. Dbpedia: A nucleus for a web of open data // The semantic web. Springer, Berlin, Heidelberg, 2007. P. 722-735.
[8] Vrandečić D., Krötzsch M. Wikidata: a free collaborative knowledgebase // Communications of the ACM. 2014. T. 57, №. 10. P. 78-85.
[9] Jiang K., Wu D., Jiang H. FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style Question-Answer Pairs with Freebase // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019. P. 318-323.

[10] SZ H. et al. CASIA@ V2: A MLN-based question answering system over linked data. 2014.
[11] Ruseti S. et al. QAnswer-Enhanced Entity Matching for Question Answering over Linked Data // CLEF (Working Notes). 2015. P. 28-35.
[12] Beaumont R., Grau B., Ligozat A. L. SemGraphQA@ QALD5: LIMSI participation at QALD5@ CLEF. 2015.
[13] Shekarpour S. et al. Sina: Semantic interpretation of user queries for question answering on interlinked data // Journal of Web Semantics. 2015. T. 30. P. 39-51.
[14] Cabrio E. et al. QAKiS: an open domain QA system based on relational patterns. 2012.
[15] Zhang Y. et al. Question answering over knowledge base with neural attention combining global knowledge information // arXiv preprint arXiv:1606.00979. 2016.
[16] Bao J. et al. Constraint-based question answering with knowledge graph // Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. 2016. P. 2503-2514
[17] Petrochuk M., Zettlemoyer L. Simplequestions nearly solved: A new upperbound and baseline approach // arXiv preprint arXiv:1804.08798. 2018.
[18] Porter M. F. Snowball: A language for stemming algorithms. 2001.
[19] Yadav V., Bethard S. A survey on recent advances in named entity recognition from deep learning models // arXiv preprint arXiv:1910.11470. 2019.
[20] Yih W. et al. The value of semantic parse labeling for knowledge base question answering // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2016. P. 201-206.
[21] Mendes P. N. et al. DBpedia spotlight: shedding light on the web of documents // Proceedings of the 7th international conference on semantic systems. 2011. P. 1-8.