СЕГМЕНТАЦИЯ ТЕКСТА В ПРОЕКТЕ «ОТКРЫТЫЙ КОРПУС»

Бочаров В. В. (bocharov@opencorpora.org),
Алексеева С. В. (alexeeva@opencorpora.org),
Грановский Д. В. (granovsky@opencorpora.org),
Остапук Н. А. (nataxan90@gmail.com),
Степанова М. Е. (mariarusia@gmail.com),
Суриков А. В. (ksurent@opencorpora.org)

Проект «Открытый Корпус» OpenCorpora.org

Сегментация текста на слова и предложения является необходимым этапом автоматической обработки текста. Последующие этапы анализа существенным образом зависят от принятых соглашений о делении на слова и предложения. В проекте Открытый корпус вручную была проведена сегментация текста объёмом более 600 тыс. текстоформ. Полученные данные были использованы в качестве обучающей выборки для создания модуля автоматической сегментации.

Ключевые слова: корпус, сегментация, токенизация, графематика, машинное обучение

TEXT SEGMENTATION IN OPENCORPORA PROJECT

Bocharov V. V. (bocharov@opencorpora.org),
Alexeeva S. V. (alexeeva@opencorpora.org),
Granovsky D. V. (granovsky@opencorpora.org),
Ostapuk N. A. (nataxan90@gmail.com),
Stepanova M. E. (mariarusia@gmail.com),
Surikov A. V. (ksurent@opencorpora.org)

OpenCorpora.org

By segmentation we mean text tokenization and sentence splitting. By default most text processing pipelines start from these two jobs before any morphology or syntax. Moreover later analysis stages depend on the way tokenization and sentence splitting are performed. In OpenCorpora project we have done manual tokenization and sentence splitting works on about 600K textforms. A machine learning technique has been trained on this data and obtained results as well as segmentation standard and its motivation are described in this article.

Key words: corpora, segmentation, tokenization, machine learning

1. Введение

Под сегментацией текста мы понимаем деление текста на слова и предложения. Этот этап является первым при решении задач автоматической обработки текста, т. е. выполняется до запуска морфологического и последующих этапов анализа. Работа последующих этапов в существенной мере зависит от того, как поставлены границы структурных единиц текста на этапе сегментации. Во многих случаях принятие решения о постановке границы в том или ином месте кажется очевидным. В то же время встречаются сложные случаи, которые заставляют задуматься о том, должна ли быть поставлена граница в данном месте или нет. Это в равной степени касается как деления текста на слова, так и на предложения. В рамках проекта OpenCorpora была проведена работа по сегментации текстов общим объёмом более 600 тыс. словоупотреблений. На полученных данных была обучена математическая модель сегментации текста на слова. В этой статье мы опишем используемые данные, требования к программному модулю сегментации, принятые в проекте OpenCorpora правила сегментации текста на структурные единицы, причины, по которым они приняты, и полученные результаты.

1.1. О проекте OpenCorpora

Целью проекта OpenCorpora является создание размеченного вручную корпуса текстов на русском языке, полностью доступного для исследователей (см. [1] [2]). При этом основными пользователями корпуса мы видим специалистов, создающих программное обеспечение для автоматической обработки текста. Мы предполагаем два возможных сценария использования: обучение математических моделей на материалах корпуса и использование корпуса как тестовой коллекции для оценки качества систем анализа текста, основанных на правилах. Оба сценария требуют, чтобы разметка была единообразной. Т.е. если в корпусе имеются несколько содержательно одинаковых примеров, то они должны быть разобраны одинаковым образом. Требование единообразия реализуется при помощи стандарта разметки, которому следуют все специалисты, размечающие тексты. Если со временем становится ясно, что стандарт необходимо изменить, то вместе с изменением стандарта изменяются и все соответствующие этому изменению примеры.

1.2. О составе корпуса

В проект OpenCorpora включаются свободно распространяемые тексты, опубликованные на условиях, совместимых с лицензией CC-BY-SA, используемой в проекте. Таковыми являются материалы, опубликованные под этой же лицензией, находящиеся в общественном достоянии и не являющиеся предметом закона об авторском праве. С одной стороны, это существенным образом сужает круг материалов, которые можно использовать, с другой — является необходимым для того, чтобы корпус целиком был свободно доступен.

Актуальный состав корпуса постоянно изменяется в связи с добавлением новых текстов и представлен на странице http://opencorpora.org/?page=stats .

2. Требования к программному модулю сегментации текста на различные структурные единицы в проекте Opencorpora

Программный модуль сегментации текста запускается первым при автоматической обработке текста и передает свои данные на последующие этапа анализа. Последующие этапы анализа могут существенным образом различаться в зависимости от конечной цели. Это может быть результат цепочки модулей лингвистического анализа текста, включающей морфологию и синтаксис, или подсчёт частот слов для построения поискового образа документа, вектора признаков, необходимых для классификации или кластеризации. Таким образом, при проектировании модуля сегментации не следует полагаться на то, что какие-то другие модули (например, модуль морфологического анализа) будут запускаться перед модулем сегментации, и, следовательно, нельзя рассчитывать на доступность результатов работы этих модулей.

При разработке модуля деления на слова мы исходили из того, что при обработке текста модули запускаются последовательно, и результаты работы более поздних этапов анализа не доступны на предыдущих этапах. Сама по себе последовательная модель обработки не является единственно возможной, однако организация двунаправленного взаимодействия модулей усложнит интерфейсы их взаимодействия и сделает эти модули менее взаимозаменяемыми, что нежелательно.

Входными данными для модуля сегментации является последовательность знаков (букв кириллицы и других алфавитов, знаков пунктуации, цифр, пробельных символов и других, предусмотренных кодировкой знаков). Таким образом, никакая лингвистическая информация по умолчанию не предполагается доступной во время сегментации.

Актуальным способом представления текста в памяти компьютера является кодирование по стандарту Unicode. И хотя не все тексты на русском языке в настоящий момент представлены таким образом, мы считаем, что программный интерфейс современных систем анализа текста должен следовать стандарту Unicode, т. к. другие часто используемые способы кодирования (cp1251, koi8-r) могут быть без потерь перекодированы в Unicode. Обратное неверно. Более подробное обсуждение формы нормализации текста, закодированного в Unicode, мы оставим за рамками этой статьи, поскольку считаем этот вопрос в большей мере техническим, чем содержательным.

На выходе модуль сегментации группирует знаки в единицы более высокого уровня. В настоящей работе единицами, на которые происходит сегментация, являются токены, предложения и абзацы. При этом предложения состоят из токенов, а абзацы — из предложений. Полученные единицы станут в дальнейшем объектами последующих уровней анализа. Токены, являющиеся словами, получат морфологическую интерпретацию. Токены, не являющиеся словами, будут отмечены соответствующим классом сущностей: знаки препинания, числа, цифробуквенные комплексы (названия моделей, мероприятий, …), интернет-адреса. Предложения станут объектом синтаксического анализа и получат интерпретацию в виде синтаксического разбора. Мы считаем, что на более высоких уровнях анализа не должна возникать необходимость дополнительно делить токены на части. Расчленить на минимальные единицы, а после — сгруппировать, если будет такая необходимость, кажется более приемлемым решением, чем решать задачу сегментации на части в различных модулях системы.

Таким образом, вопрос о делении цепочки знаков на токены становится вопросом о том, станет ли данная цепочка в дальнейшем объектом морфологического, а после лексико-сематнического анализа (если не делить) или синтаксического анализа (если поделить).

3. Правила сегментации текста на токены

В проекте OpenCorpora выполняется сегментация силами волонтёров. Для того чтобы обеспечить единообразие при сегментации разными участниками,

были разработаны правила сегментации, регламентирующие постановку границ. Далее будут описаны эти правила и причины, по которым эти они были приняты. Как уже было сказано выше, правила не являются заданными раз и навсегда. Если принимается решение изменить правила, то все соответствующие этим изменениям примеры также переразмечаются.

1. Пробелы всегда разделяют токены. Таким образом, не может быть токенов, содержащих пробел. Все цепочки, включающие пробел (например, числа, в которых пробел служит разделителем разрядов), но обозначающие одну сущность, будут объединены на последующих этапах анализа.

2. Знаки препинания, как состоящие из одного знака (точка в конце предложения и после сокращённого слова, запятая, вопросительный и восклицательный знаки, …), так и из нескольких знаков (многоточие, несколько идущих подряд одинаковых знаков препинания) являются отдельными токенами. Интерпретация многоточия как отдельного знака препинания, а не как последовательности из трёх точек, кажется целесообразным с точки зрения последующего синтаксического анализа.

3. Числа, включающие точку или запятую в качестве разделителя целой и дробной частей, обыкновенные дроби, числа со степенью, интернет-адреса (URL, e-mail, …), химические формулы считаются целыми токенами. С точки зрения лингвистического анализа текста, анализ структуры этих цепочек знаков не несёт смысла, а значение имеет только класс выделенной цепочки.

4. Имена собственные, содержащие внутри знаки препинания (“Яндекс. Деньги”, “О.С.П.-студия”, “Санкт-Петербург”, “Нью-Йорк”, …) считаются целыми токенами. Эти имена могут быть написаны самыми разными способами и могут включать не только точки и дефисы, но и другие знаки (@, #, $, …). На последующих этапах анализа кажется целесообразным интерпретировать эти имена тем же способом, как интерпретируются все другие названия городов, компаний и брендов.

5. Обозначения единиц измерения (“км”, “$”, “%”, …), а также “+” и “-” перед числами отделяются от числа и становятся отдельными токенами, даже если между ними и числом нет пробелов. На последующих этапах анализа эта конструкция будет интерпретироваться одинаково (как число с указанием единицы измерения и знака) вне зависимости от того, были ли там пробелы.

6. Частицы, написанные через дефис с другим словом (“они-то”, “подойди-ка”, “он-таки”, “точно-с”, “он-де”, ...) отделяются. Таким образом, цепочка “они-то” поделится на три токена: “они”, “-”, “то”. Это нужно для того, чтобы частица попала в поле зрения синтаксиса как отдельная единица.

7. Повторяющиеся слова, написанные через дефис (“ха-ха-ха”, “много-много”, “красный-красный”, ...) разделяются аналогично предыдущему случаю, чтобы данное явление рассматривалось при последующем синтаксическом анализе.

8. Существительные, написанные через дефис и не подпадающие под предыдущие случаи, разделяются, если словарной статьи с таким заголовком нет ни в толковом, ни в орфографическом словаре (фактически проверка выполняется по коллекции Яндекс.Словарей). Это правило введено для того, чтобы

отделить синтаксическую конструкцию приложения от составных слов, написанных через дефис.

4. Правила сегментации текста на предложения

Деление текста на предложения чаще всего очевидно: границы предложений проставляются после знаков препинания, обозначающих конец предложения (точки, восклицательного знака, вопросительного знака) перед заглавной буквой или концом абзаца. Необходимость в более детальных правилах возникает в связи с прямой речью и цитированием.

Прямая речь отделяется от слов автора, даже когда слова автора не начинаются с заглавной буквы, а в конце прямой речи нет знака препинания, обозначающего конец предложения (вертикальная черта обозначает границу предложения):

В эту минуту кашлянул ребёнок, и я различил голос Горчи, он спрашивал: |
— Ты, малый, не спишь? |
— Нет, дедушка, | *— отвечал мальчик,* | *— мне бы с тобой поговорить.* |
— А, поговорить со мной? | *А о чём поговорить?* |
(А. К. Толстой, “Семья вурдалака”, 1839)

Таким образом, выполняется деление на минимальные единицы, которые после могут быть сгруппированы в единицы более высокого уровня, если это нужно. В дальнейшем предполагается помечать, какие из выделенных предложений являются прямой речью, а какие — словами автора.

При цитировании, оформленном при помощи кавычек, границы предложений расставляются по основному правилу, т. е. по знаку препинания, обозначающему конец предложения и началу следующего предложения с заглавной буквы. При этом возможна ситуация, когда открывающая кавычка будет в одной предложении, а закрывающая — в другом:

Накалив печь в деревянных банях, «туда входили нагими и там обливались водой. | *Потом брали розги (веник) и начинали себя бить, и до того секли, что едва выходили живыми.* | *Но потом, окатившись холодной водой, оживали».* |
(“Холодная ванна возвращает силы”, “Частный корреспондент”, 2008 г., http://www.chaskor.ru/article/holodnaya_vanna_vozvrashchaet_sily_14)

Такой способ деления на предложения сохраняет синтаксически целые предложения, отодвигая на второй план указание на авторство, отмеченное при помощи кавычек. Мы считаем, что для задач лингвистического анализа первое важнее второго.

Если цитируется название художественного произведения, включающее в себя несколько предложений, то границы внутри названия не ставятся:

Роман “Опасные связи. Или письма собранные в одном частном кружке лиц и опубликованные господином Ш.де Л.в назидание некоторым другим” увидел свет в 1782 году. |

В приведённом примере название романа могло бы быть однословным, и это не повлияло бы на синтаксическую структуру предложения, если рассматривать название как единое целое.

5. Автоматическая сегментация на токены

Первоначально модуль сегментации на токены (токенизатор) был разработан для того, чтобы упростить работу волонтёров по расстановке границ. На настоящий момент он выделен в отдельный модуль на языке Perl и опубликован в библиотеке модулей CPAN [6]. Модуль токенизации разработан с использованием машинного обучения, и он переобучается при добавлении новых текстов в корпус.

В качестве обучающего множества используются все уже токенизированные вручную предложения корпуса. Каждое предложение представлено в виде пары *(s, T)*, где *s* — исходная, нетокенизированная запись предложения (строка), а *T* — упорядоченное множество токенов, на которые оно разбито. При этом *s* всегда можно получить, "склеив" все элементы *T* по порядку, вставляя пробелы, если нужно. Следовательно, для каждой позиции j в s можно определить, стоит ли после j-го символа граница токенов.

Модель представляет собой вероятностный бинарный классификатор: для каждой позиции j вычисляется некоторое количество логических контекстных функций, ее характеризующих (см. ниже), и из результатов вычисления составляется вектор, например, <0, 0, 1, 1, 0, 1>, если функций 6. Накапливая информацию о парах (вектор, наличие границы), после обработки всего обучающего множества получаем для каждого вектора оценку вероятности встретить границу токенов в данных условиях.

Таким образом, применение модели для токенизации текста выглядит так: в каждой позиции текста по тем же правилам, что и при обучении, вычислить вектор двоичных признаков, найти в таблице соответствующую вероятность и решить, достаточно ли эта вероятность высока для постановки границы в данной позиции.

Сходные модели применяются для сегментации русского текста на предложения ([3, 4]), причем во втором случае предлагается более сложный (и, по-видимому, более эффективный) метод, использующий деревья решений. Нет никаких сомнений, что этот вариант применим и в нашем случае; мы планируем исследовать его в будущем.

5.1. Контекстные функции

Большинство контекстных функций работают с минимальным, символьным контекстом — левым или правым. Сюда, например, относятся:

- "является ли символ слева буквой кириллицы",
- "является ли символ справа дефисом",
- "является ли символ справа закрывающей скобкой любого вида" и т. п.

Для работы таких функций достаточно информации, содержащейся во входном тексте.

Кроме того, некоторые функции принимают во внимание всю цепочку непробельных символов, внутри которой находятся — неважно, оказывается ли она в левом контексте, правом или обоих. Это имеет смысл, когда такая цепочка может оказаться объектом некоторого класса, который подразумевает действие “всегда разделять” или “никогда не разделять”, например:

- адресом интернет-ресурса (http://domain.com/a/b/c/page.html) или электронной почты (mail@domain.com),
- словарным словом с дефисом (*по-английски*) и т. д.

Некоторым из таких функций требуются внешние источники информации — например, список всех словарных слов с дефисом или список исключений. Таким образом, в этом месте имеется неполное соответствие требованию не использовать лингвистическую информацию на этапе сегментации. С другой стороны — полного использования морфологического словаря в данном случае так же не происходит.

5.2. Порог отсечения

Принцип работы модели подразумевает, что каждой позиции нетокенизированного текста приписывается некоторая вероятность встретить в этой позиции границу токенов. Очевидно, что для разбиения этого недостаточно, так как необходимо привести каждую вероятность к бинарному значению (разбивать или не разбивать), для чего, в свою очередь, необходимо выбрать порог отсечения — т. е. значение вероятности, начиная с которого следует считать границу “достаточно” вероятной.

Стратегия выбора порога отсечения в целом представляется понятной: необходимо подобрать такое значение, при котором токенизатор будет показывать наибольшую точность работы (в широком смысле) при разбиении предложений обучающего множества (см. раздел “Оценка качества”). Это, очевидно, эффективная стратегия в случае, если в дальнейшем предполагается полностью автоматическая работа токенизатора. Однако если по каким-то причинам важнее, допустим, максимизировать полноту за счет точности (иными словами, лучше поставить несколько лишних границ, чем пропустить границу), то порог имеет смысл снизить примерно до 0,01.

5.3. Оценка качества

Точность бинарных классификаторов (каковым является токенизатор) общепринято оценивать путем подсчета количества ошибок, которые модель допускает при классификации объектов из некоторого корпуса, при этом ошибки бывают первого рода (ложное срабатывание) и второго рода (пропуск

события). В нашем случае ошибка первого рода — неверно поставленная граница токенов, а второго — пропуск правильной границы. Если рассматривать токенизацию как задачу поиска границ, то можно воспользоваться терминологией информационного поиска и, зная количество ошибок обоих типов, вычислить точность и полноту классификатора.

При оценке мы воспользовались известной техникой кросс-валидации (cross-validation, в русскоязычной литературе встречается также перевод “скользящий контроль”) [5], которая подразумевает, что некоторое размеченное множество примеров (в нашем случае — весь доступный корпус, около 600 тыс. токенов) разбивается на N подмножеств, после чего производится N измерений, в каждом из которых очередное подмножество выступает как тестовое, а остальные в совокупности — как обучающее. Затем результат усредняется. Мы выбрали N = 10: это значение достаточно велико, чтобы усреднение имело смысл, но ещё не столь велико для того, чтобы объём тестового множества стал слишком мал.

Поскольку, как мы уже упоминали, в проекте принято соглашение, что пробел всегда является разделителем токенов, довольно бессмысленно считать все верно расставленные границы до и после пробелов. Поэтому такие случаи были исключены из оценки.

На рис. 1 представлена зависимость точности (P), полноты (R) и F1-меры от порога отсечения (шаг по оси абсцисс — 0,05, кроме того, добавлены значения 0,01 и 0,99).

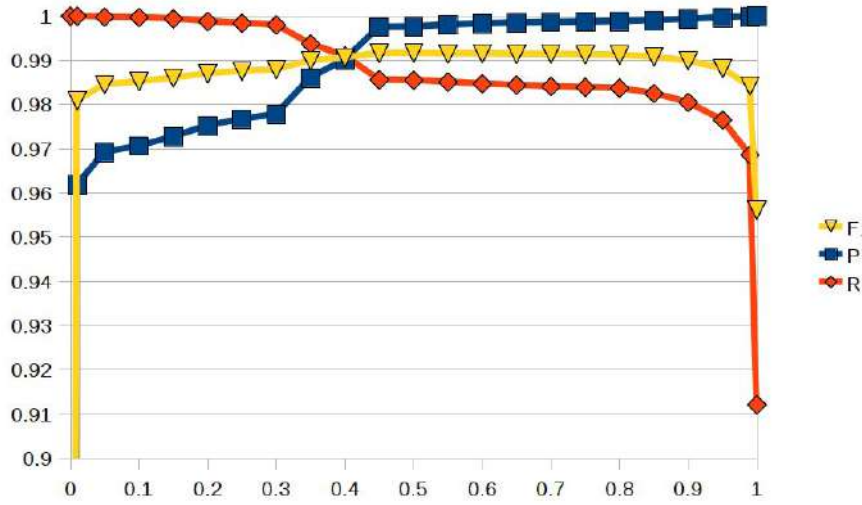

Рис. 1

Как и следовало ожидать, при увеличении порога отсечения точность монотонно растет, а полнота монотонно убывает. Интересно, что точность при пороге 0 составляет всего около 4%, а при пороге 0,01 уже более 96%. Полнота

при пороге 1 составляет около 91 %. Максимум F1-меры достигается при пороге около 0,45–0,5 и составляет примерно 99,15 %. Как показывает практика использования токенизатора в Открытом корпусе, это соответствует вполне удовлетворительному качеству токенизации, при котором большинство частотных случаев классифицируется верно. Самые спорные случаи классифицировать иначе, чем вручную, по-видимому, невозможно.

Заметим, что таким образом мы оцениваем, разумеется, не саму вероятностную модель, а конкретную ее реализацию, главным образом — набор контекстных функций, а также — в некоторой степени — качество обучающего множества.

Для сравнения мы проанализировали результат работы простейшего эвристического токенизатора с двумя правилами: всегда разбивать по пробелам, всегда разбивать на границе буквы и не буквы. F1-мера составила около 92 %.

5.4. Преимущества и недостатки метода

Описанный метод, в принципе, обладает обычным преимуществом статистических методов: существенно проще — по сравнению с моделями, основанными на правилах — менять поведение токенизатора на некотором классе случаев (путем изменения набора контекстных функций).

При этом видятся три основных недостатка:

- невысокая скорость работы как при обучении, так и при классификации, не менее чем линейно зависящая от числа функций,
- зависимость от качественных (т. е. размеченных согласно единому стандарту) обучающих примеров,
- при собственном стандарте токенизации — необходимость изначально существующего размеченного множества примеров для первичного обучения.

Очевидно, что для некоторых задач этот метод токенизации не подойдет. Мы предполагаем два сценария, в которых его применение, скорее всего, оправдано:

- разметка объемного корпуса, особенно в условиях меняющегося по ходу стандарта токенизации,
- токенизация текста для задач, где детали стандарта токенизации не очень важны (подразумевается наличие уже обученной модели, см. ниже).

6. Заключение

В настоящей статье мы рассмотрели задачу сегментации текста, включая требования к её решению и место этого решения в системе автоматической обработки текста, правила сегментации текста при создании обучающей выборки, процедуру и результаты машинного обучения. Программный модуль,

реализующий описанную модель, доступен для свободного использования всем заинтересованным исследователям и разработчикам. Обучающая выборка, использованная для создания этого программного модуля, тоже опубликована, что даёт возможность разрабатывать иные способы решения указанной задачи.

Литература

1. *Bocharov V., Bichineva S., Granovsky D., Ostapuk N., Stepanova M.* Quality assurance tools in the OpenCorpora project // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 25–29 мая 2011 г.). Вып. 10 (17). — М.: РГГУ, 2011.
2. *Бочаров В. В., Грановский Д. В.* Программное обеспечение для коллективной работы над морфологической разметкой корпуса // Труды международной конференции «Корпусная лингвистика — 2011». 27–29 июня 2011 г., Санкт-Петербург. — СПб.: С.-Петербургский гос. университет, Филологический факультет, 2011.
3. *Урюпина О.* Автоматическое разбиение текста на предложения для русского языка. // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (4–8 июня 2008 г.). Вып. 7 (14). — М.: РГГУ, 2008.
4. *Кудинов А. С., Воропаев А. А., Калинин А. Л.* Высокоточный метод распознавания концов предложений. // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 25–29 мая 2011 г.). Вып. 10 (17). — М.: РГГУ, 2011.
5. *Kohavi R.* A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection // International Joint Conference on Artificial Intelligence, 1995.
6. *Суриков А.* Модуль Lingua::RU::OpenCorpora::Tokenizer. URL: http://search.cpan.org/~ksuri/Lingua-RU-OpenCorpora-Tokenizer/