Лекция 2.3.2. Корпусная лингвистика

Под корпусной лингвистикой понимается раздел лингвистики, занимающийся разработкой и использованием лингвистических корпусов данных. Прежде, чем говорить о корпусной лингвистике, необходимо определить само понятие лингвистического корпуса. Существует довольно много определений, которые сходятся в одном: корпус есть некоторый набор текстов. Вот несколько распространённых дефиниций:

- корпус – это организованное определённым образом словесное единство, элементами которого являются тексты или специальным образом отобранные отрывки из текстов;
- корпус – это набор лингвистических данных из определённого языка в форме записанных высказываний или письменных текстов, доступный для анализа;
- корпус – это набор естественных текстов на любом языке, устных или письменных, который хранится в электронном виде и позволяет организовать компьютеризированный поиск.

В нашем курсе мы будем использовать следующее определение:

корпус – это коллекция взаимосвязанных документов (текстов) на естественном языке. Корпус может быть большим или маленьким, но обычно состоит из десятков и даже сотен гигабайт данных в тысячах документов.

В целом, корпус данных представляет собой сформированную по определенным правилам выборку данных из т.н. проблемной области, т.е., по сути, корпус данных представляет собой результат отображения проблемной области. Под проблемной областью понимается область реализаций языковой системы, содержащая феномены, подлежащие лингвистическому описанию.

Из множества определений корпуса можно выделить его главные свойства:

- электронный – в современном понимании корпус должен быть в электронном виде;
- репрезентативный – должен хорошо «представлять» объект, который моделирует;
- размеченный – главное отличие корпуса от коллекции текстов;
- прагматически ориентированный – должен быть создан под определённую задачу;

Важнейшее свойство корпуса – репрезентативность, то есть, способность отражать все свойства проблемной области. Репрезентативность определяется фонетическими, морфологическими, синтаксическими и стилевыми параметрами корпуса. Именно репрезентативность отличает корпус от простого набора текстов.

Корпус можно разбить на категории документов или на отдельные документы. Документы в корпусе могут различаться размерами, от отдельных сообщений в «Твиттере» до целых книг, но все они содержат текст (а иногда метаданные) и представляют набор связанных тем. Документы, в свою очередь, можно разбить на абзацы – смысловые единицы речи (units of discourse), которые обычно выражают одну идею. Абзацы также можно разбить на предложения – синтаксические единицы; законченное предложение структурно звучит как конкретное выражение. Предложения состоят из слов и знаков препинания – лексических единиц, которые определяют общий смысл, но гораздо более полезных в сочетаниях. Наконец, сами слова состоят из слогов, фонем, аффиксов и символов, то есть единиц, имеющих смысл, только когда они объединены в слова.

Анализ текста в некоторой степени – это разбиение больших фрагментов текста на составляющие их компоненты – уникальные слова, общие фразы, синтаксические шаблоны – с последующим применением к ним статистических механизмов. Изучая эти компоненты, можно создавать модели языка, позволяющие снабжать приложения возможностями прогнозирования. Очень скоро мы увидим, что существует множество уровней, на которых мы можем применить свой анализ, и все они связаны с одним центральным набором текстовых данных – корпусом.

История и современность

Можно сказать, что первые корпуса и корпусные методы появились задолго до возникновения корпусной лингвистики как научного направления. По сути, любое лингвистическое исследование, основанное на сопоставлении и анализе контекстов, является корпусным. Еще в конце XIX - начале XX в. в целях усовершенствования средств связи, а также в целях быстрого обучения языку создавались частотные словари. Частотность слов оценивалась по специальным выборкам текстов.

Первые корпуса в строгом смысле этого слова появились в 60-х гг. XX в. Прообразом для них послужили словарные картотеки – собрания фрагментов текстов, обычно в виде карточек, содержащих то или иное слово, и систематизированные относительно описываемого слова (в основном, по алфавиту).

В 1963 г. в Брауновском университете (США) для создания частотного словаря американского варианта английского языка был создан большой корпус на цифровом носителе (Brown Corpus), включающий 1 млн. слов. При оценке частоты некоторого слова в языке возникает проблема "сбалансированности" выборки. В языке частотность многих слов обусловлена

тематикой текстов. Так, например, слово *переменная* будет чрезвычайно частотно в математических текстах. Вероятность же встретить данное слово в художественной литературе очень мала. Для обеспечения корректности данных относительно частоты употребления слов создатели корпуса (У. Френсис и Г. Кучера) разработали строгую процедуру отбора текстов: в корпус вошли 500 фрагментов прозаических текстов, относящихся к 15 наиболее массовым жанрам и напечатанных в 1961г.

Возникновение корпусных методов связано с бурным развитием компьютерных технологий во второй половине XX в. Возможность сканирования и распознавания текста (перевод в текстовый формат), появление баз данных и систем управления базами данных сделали возможным сбор, хранение и обработку огромных массивов текстовых данных. Не последнюю роль в развитии корпусной лингвистики сыграла популяризация мировой сети Интернет, т.к. корпуса стали доступны широкому кругу пользователей, значительно расширились возможности их наполнения.

В России разработкой и исследованием корпусов занимаются специалисты Центра лингвистической документации при Независимом московском университете, отдела экспериментальной лексикографии Института русского языка им. В. В. Виноградова РАН, Института языкознания РАН, Института проблем передачи информации РАН, Всероссийского института научной и технической информации РАН, Института лингвистических исследований РАН в Санкт-Петербурге и др.

Важной вехой в развитии отечественной корпусной лингвистики явилось создание Национального корпуса русского языка. Работы по созданию Корпуса были начаты в 2001 году группой лингвистов из Москвы, Петербурга, Воронежа и других городов. В рамках развития проекта ведется работа по созданию новых ресурсов на базе корпуса.

Основные понятия корпусной лингвистики

Центральное понятие корпусной лингвистики, как мы уже выяснили – корпус. Тексты в корпус выбираются не случайным образом, а в соответствии с проблемной областью, т.е. областью реализаций интересующих исследователя языковых явлений. Проблемная область имеет два аспекта: языковой и речевой. Языковой аспект – это само изучаемое явление, а речевой – это множество контекстов, в которых это явление представлено. Проблемная область может быть как очень широкой (все произведения Достоевского Ф.М.), так и достаточно узкой (случаи согласования сказуемого с количественной группой по числу).

Одним из принципиальных вопросов является вопрос о том, какие тексты и в каком объеме необходимо отобрать в корпус. С одной стороны, хотелось бы, чтобы исследуемое явление, как бы оно ни было редко в языке, нашло отражение в корпусе. Одним из требований, предъявляемым к составу и структуре корпуса является требование полноты.

Данное требование входит в противоречие с другим важным принципом создания корпуса - требованием репрезентативности. Задача создателей корпуса – собрать как можно большее количество текстов, относящихся к тому подмножеству языка, для изучения которого корпус создается. Каким бы специфичным ни был феномен, ни один корпус не может содержать все его реализации. Поэтому корпус – это всегда определенная выборка из проблемной области, которая осуществляется на основе некоторых критериев, устанавливаемых исследователем в зависимости от задачи. Такая выборка должна отражать те или иные параметры исследуемого языкового явления в той же пропорции, что и в языке вообще или в некотором исследуемом подмножестве языка.

Предметные корпусы

Очень часто тестирование модели естественного языка начинают с использованием обобщенного корпуса. Например, есть много примеров и научных статей, в которых используются готовые наборы данных, такие как корпус Brown, корпус из «Википедии» или корпус Cornell диалогов из фильмов. Однако наиболее удачные модели языка часто являются узкоспециализированными для конкретного применения.

Почему модели языка, обученные на ограниченной предметной области, часто действуют лучше, чем такие же модели, но обученные на обобщенном корпусе? В разных предметных областях используется разный язык (словарь, сокращения, типичные фразы и т. д.), поэтому корпус, специализированный для конкретной области, анализируется и моделируется точнее, чем корпус с документами из разных областей.

Возьмем для примера термин «bank» (банк). В области экономики, финансов или политики этим термином обозначается организация, производящая фискальные и монетарные инструменты, тогда как в авиации и дорожном движении термин «bank» (крен) обозначает форму движения, которая изменяет направление воздушного судна или транспортного средства. Благодаря обучению на более узком контексте, пространство предсказаний модели становится более узким и специализированным, а значит, такая модель лучше справляется с гибкостью языка. Наличие предметного корпуса имеет

большое значение для создания удачного приложения данных на основе анализа естественного языка.

Часто специалисты по анализу и обработке данных начинают с создания единого статического набора документов и затем применяют определенные виды анализа. Однако независимо от процедуры анализа и приемов получения данных модель в этом случае получится статичной, не способной реагировать на новую обратную связь и не отражающей динамическую природу языка.

Структура корпуса

Самый простой и распространенный способ организации текстового корпуса для управления им – хранение документов в файловой системе на диске. Организуя размещение документов в корпусе по подкаталогам, можно обеспечить их классификацию и содержательное разделение по имеющейся метаинформации, такой как даты. Благодаря хранению каждого документа в отдельном файле, инструменты чтения корпуса могут быстро отыскивать разные подмножества документов, обработка которых может осуществляться параллельно, когда каждый процесс получает свое, отличное от других подмножество документов.

Текст также является наиболее сжимаемым форматом, что делает zip-файлы со структурой каталогов на диске идеальным форматом для передачи и хранения данных. Наконец, корпусы, хранящиеся на диске, обычно статичны и обрабатываются как единое целое.

Однако хранение документов в отдельных файлах может приводить к проблемам. Небольшие документы, такие как электронные письма или сообщения из «Твиттера», не имеет смысла хранить в отдельных файлах. Кроме того, электронные письма обычно хранятся в формате MBox – обычном текстовом формате с разделителями, разделяющими разные составные части сообщений, содержащие простой текст, HTML, изображения и вложения. Обычно такие документы можно организовать по категориям, имеющимся в почтовой службе (входящие, помеченные, архивные и т.д.).

Твиты, как правило, – это небольшие структуры данных в формате JSON, включающие в себя не только текст твита, но и другие метаданные, такие как имя и местоположение пользователя. Обычно для хранения сразу нескольких твитов используется формат JSON с символами перевода строки в качестве разделителя, который иногда называют форматом строк JSON. Этот формат позволяет легко читать твиты по одному и выполнять парсинг построчно, а также отыскивать разные твиты в файле. Если все твиты хранить в одном файле, размер такого файла может получиться очень большим, поэтому часто применяется организация твитов в файлы по именам пользователей,

местоположению или дате, что позволяет уменьшить размеры отдельных файлов и создать осмысленную структуру каталогов.

Также для сохранения данных с некоторой логической структурой нередко используется прием ограничения размеров файлов. Например, запись данных в файл (с учетом границ между документами) производится до достижения некоторого предела (например, 128 Мбайт), после чего открывается новый файл и запись продолжается в него.

Обычно корпус можно представить в виде древовидного каталога:

```
corpus
├── citation.bib
├── feeds.json
├── LICENSE.md
├── manifest.json
├── README.md
└── books
|  ├── 56d629e7c1808113ffb87eaf.html
|  ├── 56d629e7c1808113ffb87eb3.html
|  └── 56d629ebc1808113ffb87ed0.html
└── business
|  ├── 56d625d5c1808113ffb87730.html
|  ├── 56d625d6c1808113ffb87736.html
|  └── 56d625ddc1808113ffb87752.html
└── cinema
|  ├── 56d629b5c1808113ffb87d8f.html
|  ├── 56d629b5c1808113ffb87d93.html
|  └── 56d629b6c1808113ffb87d9a.html
└── cooking
 ├── 56d62af2c1808113ffb880ec.html
 ├── 56d62af2c1808113ffb880ee.html
 └── 56d62af2c1808113ffb880fa.html
```

Это корпус HTML записанный на диск. По структуре каталогов и именам файлов легко определить, какие являются документами, а какие хранят метаинформацию. Структурированный подход к управлению и хранению корпуса обеспечивает следование текстового анализа правилу воспроизводимости, способствуя улучшенной интерпретации и уверенности в результатах.

Объекты чтения корпусов

После структурирования и организации корпуса на диске появляются две возможности: использовать систематический подход к чтению корпуса в

программном контексте, а также следить за изменениями в корпусе и управлять ими.

Большинство нетривиальных корпусов содержит тысячи документов с общим объемом текстовых данных, порой достигающим нескольких гигабайт. Исходные строки текста, загруженные из документов, необходимо подвергнуть предварительной обработке и преобразовать в представление, пригодное для дальнейшего анализа. Это аддитивный процесс, в ходе которого могут генерироваться и дублироваться данные, что увеличивает объем необходимой для работы памяти. Это важный с вычислительной точки зрения аспект, потому что без некоторого метода выбора документов на диске и потоковой передачи информации анализ текста будет ограничен быстродействием одного компьютера, препятствуя возможности создания интересующих нас моделей. К счастью, в библиотеке NLTK имеются хорошо продуманные инструменты потокового доступа к корпусу на диске, которые передают корпус в Python через объекты CorpusReader.

CorpusReader – это программный интерфейс для чтения, поиска, потоковой передачи и фильтрации документов, а также для предоставления возможности преобразования и предварительной обработки данных программному коду, которому требуется доступ к данным в корпусе. При создании экземпляра CorpusReader ему передается путь к корневому каталогу с файлами корпуса, сигнатура для распознавания названий документов, а также кодировка файлов (по умолчанию используется UTF-8).

Поскольку кроме документов для анализа в корпусе, чаще всего, имеются дополнительные файлы (например, README, citation, license и др.), объект чтения нуждается в некотором механизме, помогающем точно идентифицировать, какие документы являются частью корпуса. Роль такого механизма играет параметр. В нем можно указать точный перечень имен или регулярное выражение для сопоставления со всеми документами в корневом каталоге (например, `\w+\.txt`, которому соответствует одна или несколько букв или цифр в имени файла, за которыми следует расширение `.txt`). Например, в следующем каталоге этому регулярному выражению будут соответствовать три файла в каталоге speeches и файл `transcript.txt`, но не файлы license, README, citation и metadata:

```
corpus
├── LICENSE.md
├── README.md
├── citation.bib
├── transcript.txt
```

```
└── speeches
    ├── 04102008.txt
    ├── 10142009.txt
    ├── 09012014.txt
    └── metadata.json
```

Эти три простых параметра дают объекту CorpusReader возможность перебрать все документы в корпусе, открыть каждый с использованием правильной кодировки и позволяют программистам получить доступ к метаданным отдельно. Сама по себе идея CorpusReader может показаться не особенно эффектной, но при работе с множеством документов этот интерфейс позволяет программистам прочитать один или несколько документов в память, отыскать определенные места в корпусе, не открывая и не читая ненужные документы, передать данные в процесс анализа, храня в памяти документы по одному, и отфильтровать или выбрать конкретные документы из корпуса.

Потоковый доступ к данным с помощью NLTK

В состав библиотеки NLTK входит большое разнообразие объектов для чтения корпусов, предусматривающих доступ к текстовым корпусам и лексическим ресурсам, которые можно загрузить с ее помощью. Она также включает в себя универсальные вспомогательные объекты CorpusReader, которые предъявляют более жесткие требования к структуре корпуса, зато дают возможность быстро создавать корпусы и связывать их с объектами чтения. Кроме того, их имена подсказывают, для каких целей их можно использовать. Вот некоторые из наиболее примечательных объектов CorpusReader:

- PlaintextCorpusReader – предназначен для чтения корпуса простых текстовых документов, где, как предполагается, абзацы отделены друг от друга пустыми строками.
- TaggedCorpusReader – предназначен для чтения корпуса с маркерами частей речи, где каждое предложение находится в отдельной строке, а лексемы разделены соответствующими им тегами.
- BracketParseCorpusReader – предназначен для чтения корпуса, состоящего из деревьев синтаксического анализа, заключенных в круглые скобки.
- ChunkedCorpusReader – предназначен для чтения фрагментированного корпуса (возможно, размеченного тегами), отформатированного с помощью круглых скобок.
- TwitterCorpusReader – предназначен для чтения корпуса, состоящего из твитов, сериализованных в формат строк JSON.

- WordListCorpusReader – предназначен для чтения списка слов, по одному в строке. Пустые строки игнорируются.
- XMLCorpusReader – предназначен для чтения корпуса, состоящего из XML-документов.
- CategorizedCorpusReader – подмешиваемый класс для объектов чтения корпусов, в которых документы организованы по категориям.

Объекты чтения размеченных и фрагментированных корпусов, а также корпусов, состоящих из деревьев синтаксического анализа, предназначены для чтения аннотированных корпусов; собираясь вручную осуществлять предметно-ориентированное аннотирование перед машинным обучением, вы обязательно должны изучить распознаваемые ими форматы. Объекты чтения корпусов с твитами, документами XML и простыми текстовыми документами позволяют указать, как обрабатывать данные, хранящиеся на диске в разных форматах, что дает возможность создавать расширения для чтения корпусов в формате CSV, JSON или даже из базы данных.

Обработка файлов корпуса. Регулярные выражения

Рассмотрим простой корпус киносценариев сериалов Star Wars («Звездные войны») и Star Trek («Звездный путь») со следующей организацией:

```
corpus
├── LICENSE
├── README
└── Star Trek
|  ├── Star Trek - Balance of Terror.txt
|  ├── Star Trek - First Contact.txt
|  ├── Star Trek - Generations.txt
|  ├── Star Trek - Nemesis.txt
|  ├── Star Trek - The Motion Picture.txt
|  ├── Star Trek 2 - The Wrath of Khan.txt
|  └── Star Trek.txt
└── Star Wars
|  ├── Star Wars Episode 1.txt
|  ├── Star Wars Episode 2.txt
|  ├── Star Wars Episode 3.txt
|  ├── Star Wars Episode 4.txt
|  ├── Star Wars Episode 5.txt
|  ├── Star Wars Episode 6.txt
|  └── Star Wars Episode 7.txt
└── citation.bib
```

Для доступа к данным в киносценариях отлично подойдет CategorizedPlaintextCorpusReader, потому что все документы хранятся в простых текстовых файлах и разбиты на две категории, а именно: Star Wars и Star Trek. Чтобы задействовать CategorizedPlaintextCorpusReader, нужно указать регулярные выражения, которые позволят объекту чтения автоматически определять идентификаторы файлов (fileids) и категории (categories):

```
from nltk.corpus.reader.plaintext import CategorizedPlaintext
CorpusReader

DOC_PATTERN = r'(?!\.)[\w_\s]+/[\w\s\d\-]+\.txt'
CAT_PATTERN = r'([\w_\s]+)/.*'

corpus = CategorizedPlaintextCorpusReader(
 '/path/to/corpus/root', DOC_PATTERN, cat_pattern = CAT_PATTE
RN
)
```

Регулярное выражение с шаблоном документа определяет имена файлов документов как возможно включающие путь, относительно корневого каталога корпуса состоящий из одной или нескольких букв, цифр, пробелов или подчеркиваний, за которыми следует слеш (/), затем одна или несколько букв, цифр, пробелов или дефисов, и расширение .txt. Это выражение соответствует таким именам файлов с документами, как Star Wars/Star Wars Episode 1.txt, но не соответствует файлу episode.txt. Регулярное выражение с шаблоном категории является усеченной версией первого регулярного выражения и состоит из сохраняющей группы, которая определяет название категории по имени каталога (например, для Star Wars/anything.txt будет определена категория Star Wars). Начнем исследование данных на диске с изучения полученных имен:

```
corpus.categories()
# ['Star Trek', 'Star Wars']
corpus.fileids()
# ['Star Trek/Star Trek -
 Balance of Terror.txt', # 'Star Trek/Star Trek -
 First Contact.txt', ...]
```

Регулярные выражения могут быть очень сложными, но они предлагают мощный механизм, помогающий точно определить, что должен загрузить объект чтения корпуса и как. Также можно явно передать список категорий и идентификаторов файлов, но это сделает объект чтения менее гибким. Использование регулярных выражений позволяет добавлять новые категории, просто создавая новые каталоги в корпусе, и новые документы, перемещая файлы в требуемый каталог.