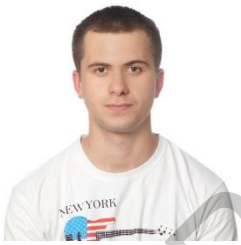
ФУНКЦИОНАЛЬНОСТЬ СИСТЕМЫ ПОЛУЧЕНИЯ И АНАЛИЗА ТЕКСТОВЫХ ДАННЫХ

М.В. Стержанов
Студент кафедры информатики БГУИР

Н.Н. Шинкевич
Студентка кафедры информатики БГУИР



М.И. Селюк
Студент кафедры информатики БГУИР

Д.Н. Рожков
Студент кафедры информатики БГУИР

В.Ю. Пресняцкий
Студент кафедры информатики БГУИР

А.И. Свито
Студент кафедры информатики БГУИР

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
E-mail: sterjanov@bsuir.by, sn0wf1llin@gmail.com, max.selyuk@gmail.com, rdimon2912@gmail.com, presniatski@gmail.com, alexandervirk@gmail.com

Abstract. The proliferation of textual data in business is overwhelming. While the amount of textual data is increasing rapidly, businesses' ability to summarize, understand, and make sense of such data for making better business decisions remain challenging. This paper describes a system that organizes and analyzes textual data for extracting insightful customer intelligence from a large collection of documents and for using such information to improve business operations and performance.

Система представляет собой современный программный комплекс с интеллектуальной функцией анализа текстового контента, которая позволяет рассчитывать статистические характеристики текста, а также автоматически выделять ключевые слова. В разработке автоматизированной системы использованы устоявшиеся методы математического моделирования и искусственного интеллекта (в частности – аппарат искусственных нейронных сетей), математической статистики, информационных технологий и программирования.

При разработке архитектуры системы в первую очередь ставилась задача упрощения автоматического тестирования большого количества алгоритмов получения и обработки данных. Поэтому система состоит из набора отдельных слабосвязанных компонентов:

- автоматическое получение данных из внешних источников;
- подсчет статистических характеристик полученного контента;
- выделение ключевых слов.

В представленной системе используются сторонние библиотеки: NLTK WordPunktTokenizer и PunktSentenceTokenizer - для выделения предложений и слов, NLTK NaiveBayesClassifier - реализация наивного Байесовского классификатора.

Данные для обработки получаются в режиме реального времени из внешних источников, в качестве которых выступают информационно-новостные сайты.

Инструменты, позволяющие собирать данные для исследований из Веба, называются «веб-пауками» (web-spider), краулерами (web crawler) или скребками (web scraper). Поисковый робот — программный комплекс, осуществляющий навигацию по веб-ресурсам и сбор информации для базы данных приложения-агента [1]. Работу разработанного краулера можно описать следующим образом: сканирование сайта начинается с начальной страницы и затем робот использует ссылки, размещенные на ней, для перехода на другие страницы. Каждая страница сайта анализируется на наличие требуемой информации, которая копируется в соответствующее хранилище в случае обнаружения. Процесс повторяется до тех пор, пока не будет проанализировано требуемое число страниц либо пока не будет достигнута некая цель. Модуль получения данных разработан на языке программирования Ruby и состоит из трех основных частей: блок сканирования и обработки данных, блок управления краулером (команды вводятся через консоль) и база данных. Собираемая роботом информация состоит из ссылочной структуры обрабатываемого ресурса и веб-страниц. В качестве основы для базы данных была выбрана бесплатная СУБД MySQL. Для упрощения взаимодействия с БД нами используется библиотека Sequel, позволяющая представлять данные в виде объектов.

Кроме непосредственной информации, полученной с помощью краулеров, для полученного контента нами также был реализован подсчет различных статистических характеристик. Нами подсчитывается:

- общее число уникальных слов;
- общее число вопросительных предложений;
- общее число заголовков, содержащих слово Why;
- общее число заголовков, имеющих определённый контекст (проверка слова на вхождение в специальный словарь);
- общее число заголовков, содержащих числовые данные;
- наиболее часто встречающиеся слова (слово, число повторений, %);
- наиболее часто встречающиеся фразы из 2 слов (фраза, число повторений, %);
- наиболее часто встречающиеся фразы из 3 слов (фраза, число повторений, %).

В базе данных сохраняется статистика, собранная в ходе каждого запуска. В дальнейшем эта информация может быть использована как признаки (features) для алгоритмов машинного обучения.

Автоматическая классификация текста является ярким примером задач, для которых довольно сложно получить непротиворечивое, достаточно представительное обучающее множество, и в то же время, сравнительно легко собрать большой объем неразмеченных документов.

Для решения подобной задачи существует два основных подхода:

1 основанные на правилах;

2 основанные на машинном обучении - использовании статистических данных, полученных из обучающей выборки.

Подходы, основанные на правилах, в настоящее время редко используются из-за сложности, возникающей при создании правил: использование узкоспециализированных лингвистических знаний, создание ряда нетривиальных правил и невозможность обобщения результатов на другие языки.

Более перспективными являются методы машинного обучения, требующие для своей работы размеченной коллекции документов.

В данной работе в качестве коллекции документов была взята база данных статей, каждая из которых содержит название, аннотацию и содержание.

Нами было реализовано разделение коллекции документов на заданное количество тем (кластеров) с применением алгоритма латентного размещения Дирихле (Latent Dirichlet Allocation, LDA). LDA принимает набор документов, в качестве которых выступает контент

статей из коллекции документов, и выдает список тем в этих документах. Каждая тема характеризуется распределением используемых в ней ключевых слов. LDA на тренировочных данных выявляет список тем, и затем для каждой статьи можно получить вероятности отношения контента данной статьи к выявленным темам. Тема, вероятность отношения контента к которой наибольшая, выбирается в качестве темы документа. Таким образом, получается распределение статей в коллекции по темам.

Для реализации данного алгоритма использовался пакет gensim и встроенный модуль gensim.models.ldamulticore.LdaMulticore, для него был написан класс-обертка, осуществляющий подготовку текста для последующего анализа, включающий перевод текста в нижний регистр, удаление стоп-слов, пунктуации, ссылок, разбиение текста на слова, а так же задающий количество тем и слов, впоследствии используемых для описания каждой темы.

Нами реализовано выделение ключевых слов из названия, аннотации и содержания документа при помощи методов Textrank, Rake, TF-IDF для последующего сравнения и анализа.

Алгоритм Textrank основан на представлении текста в виде графа. Вершины графа – целостные части текста (отдельные слова, n-граммы, предложения). Веса дуг графа характеризуют тип связи между вершинами по выбранному принципу (например, встречаться вместе в окне размера n, т.е. на расстоянии не более n слов друг от друга). В качестве вершин графа рассматриваются отдельные слова текста; вес дуги, соединяющей две вершины-слова, показывает, сколько раз эти два слова встретились в тексте в окне n. Для оценки веса каждой вершины-слова в используется величина, основанная на модификации формулы PageRank:

$$TR(t_i) = (1-d) + d \cdot \sum_{t \in In(t_i)} \frac{w_{ji}}{\sum_{t_k \in Out(t_j)} w_{jk}} \cdot TR(t_j), \quad (1)$$

где *d* - фактор затухания, *In(t)* - множество вершин входящих в t, *Out(t)* - множество вершин исходящих из *t*, *Wij* - вес ребра *ij*.

В качестве веса ребра использовалось расстояние Левенштейна между двумя отдельными словами. В качестве результата берутся *n* слов, имеющих наибольшие значения *TR*.

Метод Rapid Automatic Keyword Extraction (RAKE) основывается на том, что ключевые слова включают в себя значимые слова, но редко включают стоп-слова, местоимения или другие слова с минимальным лексическим значением.

Извлечение ключевых слов происходит следующим образом: текст разбивается на слова по позициям стоп-слов и знаков препинания - разделителей, образуя последовательности из разделителей и собственно слов, те последовательности, которые не имеют разделителей в своем составе формируют список “кандидатов” в ключевые слова. Далее строится граф встреч данных кандидатов друг с другом в тексте документа. Вычисляется вес каждого слова как отношение

$$weight(w) = \frac{\deg(w)}{freq(w)}, \quad (2)$$

где deg(w) - word degree, freq(w) - word frequency. N слов, имеющих наибольший вес, выбираются в качестве ключевых.

Алгоритм TF-IDF[4] основан на метрике tf-idf, которая рассчитывается для каждого конкретного слова в каждом документе как произведение частоты слова в данном документе tf на инвертированную частоту документов idf, где idf определяется как

$$idf = log \frac{|N|}{df}, \quad (3)$$

где *N* – множество документов, *df* – число документов, в которых хотя бы раз встретилось слово. С помощью TF-IDF оценивается вес каждого слова в документе.

В качестве группы документов мы выбираем 300 случайных статей из коллекции.

Направлением дальнейших исследований является использование полученных статистических характеристик, выделенных ключевых слов и тем в качестве исходных данных для решения задачи предсказания различных атрибутов статей методами машинного обучения.

Литература

[1]. A.H.F. Laender, A brief survey of web data extraction tools // A.H.F. Laender et al. // ACM SIGMOD Record 31(2), pp 84-93. 2002.

[2]. Blei, D.M. Latent Dirichlet Allocation / D.M. Blei, A.Y. Ng, M.I. Jordan // Journal of Machine Learning Research. — 2003. —Vol. 3. — PP. 993 — 1022.

[3]. Agirre, Eneko and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In Proc. of the 12th Conference of the European Chapter of the ACL, pages 33–41.

[4]. Aizawa, Akiko N. 2003. An information-theoretic perspective of tf-idf measures. Information Processing Management, 39(1):45–65.