О хранении графа социальной сети*

***М.И. Коломейченко*[♮], *И.В. Поляков*[‡], *А.А. Чеповский*[‡], *А.М. Чеповский*[‡]**
achepovskiy@hse.ru
[♮]Москва, Федеральный исследовательский центр «Информатика и управление» Российской академии наук
[‡]Москва, Национальный исследовательский университет «Высшая школа экономики»

Рассматривается задача анализа графа социальной сети. Представлена специализированная структура данных, предназначенная для хранения и обработки графов социальных сетей больших объемов. Предложена архитектура хранилища графа социальной сети больших объемов.

Ключевые слова: *анализ социальной сети, хранилище графов, большие данные.*

About the warehouse of social networks graph*

***M. I. Kolomeychenko*[♮], *I. V. Polyakov*[‡], *A.A. Chepovskiy*[‡], *A.M. Chepovskiy*[‡]**
[♮]Federal Research Center "Informatics and Control" of Russian Academy of Sciences, Moscow, Russia
[‡]National Research University The Higher School of Economics, Moscow, Russia

This article describes the problem of social network graph analysis. In this paper special data structure for big social graph storing and operating is presented. The architecture of warehouse of the big social network graph is offered.

Keywords: *social network analysis, graph warehouse, big data.*

Введение

Анализ социальных сетей является актуальной и сложной задачей информационных технологий [1, 2]. Многие задачи исследования социальных сетей сводятся к задачам анализа графов, полученных из социальных сетей [3, 4]. При этом важен не только сам граф, но и вся сопутствующая информация по вершинам и ребрам этих графов (пользователи сети и их контакты). В качестве примеров можно выделить следующие задачи: Анализ источников возникновения некоторой информации; Поиск узлов с наивысшими показателями цитирования по заданной рубрике (индекс цитирования – количество «репостов» по всем постам данного пользователя имеющим заданную рубрику); Определение путей и каналов распространения информации по заданной рубрике. Возникает необходимость в анализе не только графа связей объектов социальной сети, но и текстовой информации, для обработки которой необходимо использовать методы компьютерной лингвистики [5]. Для решения задач анализа необходимо создавать хранилища такой информации, взятой из социальных сетей.

Базы данных для хранения и обработки графов имеют свои особенности в зависимости от их предназначения [6]. Хранилища для хранения и обработки графов общего предназначения поддерживают большой диапазон различных операций [6]. Поэтому для конкретных предметных областей и задач могут быть построены более эффективные решения [4]. В [4, 7] мы рассматривали хранение графа связей пользователей социальной сети. Соответственно в [4, 7] были разработаны алгоритмы прокладки маршрутов в таких графов с учетом их больших размеров. В [3, 8] были разработаны алгоритмы анализа структуры таких графов.

В данной работе мы рассматриваем хранилище информации, которую можно получать посредством официальных программных интерфейсов из различных социальных сетей. Данное хранилище должно с одной стороны быть универсальным по отношению к различным вариантом информации разных социальных сетей. С другой стороны, оно должно быть специализированной базой данных для хранения и обработки графов, что обеспечивает поддержку более сложных операций в рамках моделей исследуемых предметных областей (социальных сетей в данном случае).

Структура и тип хранимой информации

Рассмотрим хранение данных, полученных в результате загрузки профилей социальных сетей, а также связей различного типа между ними.

Предполагается сохранять и анализировать информацию следующего типа:

— Профили пользователей, со всеми текстовыми атрибутами (ФИО, Образование, Родной город и прочее);
— Группы пользователей с информацией о составе участников;
— Содержание стен пользователей и групп (посты и «лайки» на них со стороны других пользователей);

Работа выполнена и опубликована при финансовой поддержке РФФИ, гранты 15-07-20370, 16-07-00641.

— Информацию об имеющихся взаимосвязях между пользователями (факт дружбы между пользователями, «лайки» от одного пользователя к другому);
— Взаимосвязи между пользователями и постами (каждый пост создан некоторым пользователем или группой и может являться или не являться «репостом» некоторого другого поста, кроме того пользователи имеют возможность ставить «лайки» и комментировать посты).

Дополнительная обработка специализированными программными модулями текстов постов, как текстов на естественном языке [5], позволяет в автоматическом режиме присваивать дополнительные атрибуты: Язык поста; Рубрика поста; Длина поста или класс поста по длине (короткий текст, длинный текст), который может определяться некоторым пороговым значением.

Таким образом в графе присутствуют объекты трех типов:

1. Пользователи (атрибуты: идентификатор социальной сети, ФИО и прочие доступные атрибуты профиля);
2. Группа пользователей (атрибуты: идентификатор социальной сети, название);
3. Посты (атрибуты: текст, количество «репостов», количество «лайков», количество комментариев).

Предполагается, что набор атрибутов для каждого типа объектов фиксируется на этапе создания хранилища. Предусмотрено хранение связей между двумя пользователями (группами) следующих типов:

— Связь дружбы между пользователями (без атрибутов);
— Факт «лайка» или комментирования одним пользователем поста другого пользователя (тип события: «лайк» или комментарий, время события, если оно доступно или же время размещения самого поста, атрибуты поста такие как язык и рубрика);
— Факт отметки одним пользователем другого пользователя в одном из своих постов (время, атрибуты поста);
— Факт «репоста» одним пользователем поста другого пользователя (время, атрибуты поста такие как язык и рубрика).
— Факт членства пользователя в некоторой группе (без атрибутов);
— Факт лайка или комментирования пользователем поста в группе (время «лайка», если оно доступно или же время размещения самого поста, атрибуты поста);
— Факт «репоста» пользователем поста группы (время , атрибуты поста);
— Факт «репоста» одной группой поста другой группы (время, атрибуты поста).

В целом группы всегда выступают в том же качестве что и пользователи и кроме того могут содержать в себе других пользователей.

Для хранимых связей между постами предусмотрены следующие типы:

— Факт состоящий в том, что один пост является «репостом» другого поста;
— Факт состоящий в том, что один пост является «репостом» комментария к другому посту.

Для хранимых связей между пользователями (группами) и постами предусмотрены следующие типы:

— Пользователь поместил данный пост у себя на стене (время);
— Пользователь «зарепостил» данный пост (время);
— Пользователь поставил «лайк» или прокомментировал данный пост;
— Пользователь был отмечен на данном посте;
— Группа поместила данный пост у себя на стене (время);
— Группа «зарепостила» данный пост (время);
— Группа поставила «лайк» или прокомментировала данный пост.

Структура хранилища определяется необходимостью поддерживать хранение двух составных частей хранилища: структур для хранения вершин графа и структур для хранения связей графа.

Хранение вершин графа

Предполагается что множество возможных типов атрибутов фиксировано на этапе создания хранилища:

$$A = \{A_1, A_2, ..., A_M\}. \quad (1)$$

Для хранения вершин предполагается использовать специализированное хранилище, состоящее из файлов $F = \{F_1, F_2, ..., F_N\}$, число которых также фиксируется на этапе создания. При этом определяется функция, которая задает типы атрибутов, сохраняющихся в заданном файле:

$$G : F \to 2^A. \quad (2)$$

Данная схема позволяет хранить атрибуты небольшого размера отдельно от значительных по объему текстовых полей, что позволяет, ускорить доступ к нетекстовым атрибутам. Возможна ситуация дублирования, когда несколько копий одного и того же

атрибута сохраняется в различных файлах. Указанная ситуация возникает в случае, если для большого числа вершин требуется обеспечивать быстрый доступ как ко всем атрибутам, так и к нетекстовым атрибутам в отдельности. В такой ситуации можно хранить все атрибуты в первом файле, продублировав дополнительно нетекстовые атрибуты во втором. Те атрибуты, которые по каким-либо причинам могут запрашиваться отдельно от остальных также могут быть продублированы (например, идентификаторы социальной сети).

Каждый файл представляет из себя набор записей переменной длины

$$F_i = \bigcup_{j=1}^{N} R_i^j. \tag{3}$$

Каждой вершине графа и каждому файлу $f \in F$ соответствует список записей $R(v, f)$ из данного файла. Записи организованы в однонаправленный список от последней к первой (обратно их расположению в файле). При добавлении очередной вершины она получает в каждом файле $f \in F$ список, состоящий из одной записи, в которую направляются все ее атрибуты из множества $G(f)$.

В дальнейшем если требуется изменить какой-либо атрибут $a \in A$ уже добавленной вершины v, следует определить список файлов, используемых для хранения данного атрибута. После чего в каждый из них добавляется новая запись, содержащая новое значение атрибута a для вершины v. Затем в каждом из файлов новая запись присоединяется к списку $R(v, f)$. Старые значения при этом сохраняются, однако должны игнорироваться при чтении списка $R(v, f)$.

Такая организация предполагает необходимость периодической очистки файлов из множества F от уже перезаписанных атрибутов, однако изменение любого значения происходит за одну дисковую операцию. При этом добавление данных в конец файла может быть эффективно буферизовано, что позволяет понизить его среднюю стоимость до очень низких величин (запись на диск выполняется один раз при каждом полном заполнении буфера). Схема является особенно удачной, если операция изменения значения атрибута у вершины выполняется достаточно редко.

Помимо списков записей в файлах из F каждая вершина имеет в оперативной памяти дескриптор $D(v)$, содержащий смещения последних записей списков $R(v, f)$, $f \in F$, а также специальный бит сигнализирующий о том, что вершина была удалена. Удаление вершины производится путем выставления данного бита в ее дескрипторе. Такой механизм также требует периодической очистки, от удаленных вершин, однако позволяет избежать неконтролируемых по сложности операций перестроения хранилища при удалении одной вершины.

По каждому атрибуту $a \in A$, создается индекс, позволяющий по заданному значению X определить список вершин, имеющих атрибут a со значением X. Кроме того, для текстовых атрибутов формируется полнотекстовый индекс с использованием автоматического анализа морфологии русского языка, который позволяет по заданному слову w определить список вершин, имеющих атрибут a, содержащий w.

Структура используемых индексов описана в [4].

Хранение связей графа

Каждая связь l имеет атрибут $a_T(l)$, задающий ее тип. Множество значений данного атрибута обозначим через T. Вершины, соединяемые данной связью, обозначим через $v_1(l)$, $v_2(l)$. При хранении связей также используется схема с дублированием. Используемое хранилище состоит из файлов, число которых фиксируется на этапе создания:

$$L = \{L_1, L_2, ..., L_K\} \tag{4}$$

При создании хранилища задается функция, описывающая файлы, в которых следует дублировать связь заданного типа:

$$G_* : T \to 2^L \tag{5}$$

Каждый файл из L представляет из себя список блоков $B_1, B_2, ..., B_j$ фиксированного размера. Для адресации блока достаточно указать его порядковый номер. Каждой вершине графа и каждому файлу $f \in L$ соответствует список блоков $B(v, f)$ из данного файла. Индекс последнего блока в каждом из списков $B(v, f)$ добавляется в дескриптор $D(v)$. Если один из списков оказался пустым, индекс не добавляется. Для обеспечения эффективного добавления данных, последний блок каждого списка должен быть кэширован в оперативной памяти, что позволит изменять его данные, не обращаясь к дисковым операциям.

При добавлении очередной связи l, определяется ее тип $a_T(l)$, по которому формируется список файлов $G_*(a_T(l))$. Для каждого $f \in G_*(a_T(l))$, информация о наличии связи сохраняется в последних блоках списков $B(v_1(l), f)$,$B(v_2(l), f)$,.

Кэширование этих блоков позволяет в большинстве случаев обходиться без обращения к дисковой подсистеме. В случае если какой-либо из блоков

окажется заполненным, к соответствующему списку следует добавить новый блок, который должен заместить в кэше заполненный. При этом помимо естественного дублирования по каждой из вершин, возможно дополнительное дублирование, создаваемое с целью ускорения работы со связями конкретного типа, которое определяется конкретным видом функции [5].

Заключение

Предложенная структура хранилища графа социальной сети предлагает достаточно гибкую схему хранения данных, загруженных из различных социальных сетей. Операции изменения атрибутов вершин, добавления очередной вершины или связи хорошо буферизованы и в среднем занимает постоянное время. При использовании простейших серверных систем хранилище предложенной структуры позволяет хранить графы, содержащие сотни миллионов объектов, при этом максимальное количество связей ограничено только объемом используемой дисковой подсистемы.

В целом предложенная структура хранилища позволяет решать широкий спектр таких задач, как анализ структуры графа, детектирование попыток информационного воздействия и изучением путей распространения информации в социальной сети.

Литература

[1] Базенков Н. И., Губанов Д. А. *Обзор информационных систем анализа социальных сетей* //Управление большими системами: сборник трудов. 2013. С.357-394.

[2] Батура Т.В. *Методы анализа компьютерных социальных сетей* //Вестник НГУ. Серия: Информационные технологии. – 2012. – Том 10. – Вып. 4. – С.13-28.

[3] Коломейченко М. И., Чеповский А. М. *Визуализация и анализ графов больших размеров* //Бизнес-информатика. – 2014. – №4(30). – С.7-16.

[4] Поляков И.В., Чеповский А.А., Чеповский А.М. *Хранение и обработка графа социальных сетей* //Вестник Новосибирского государственного университета. Серия: Информационные технологии. - 2013. - Т.11. - №4. - С.77-83.

[5] Чеповский А.М. *Информационные модели в задачах обработки текстов на естественных языках.* Второе издание, переработанное. / А.М.Чеповский - М.: Национальный Открытый Университет «ИНТУИТ», 2015. - 159 с.

[6] Angles R. A *Comparison of Current Graph Database Models* //Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW '12, IEEE Computer Society. Washington, DC, USA, 2012. pp.171–177.

[7] Polyakov I.V., Chepovskiy A.A., Chepovskiy A.M. *Algorithms for Searching Paths in Huge Graphs* //Journal of Mathematical Sciences. 2015. Vol. 211. No. 3. pp.413-417.

[8] Kolomeychenko M. I., Chepovskiy A.A., Chepovskiy A.M. *An Algorithm for Detecting Communities in Social Networks* //Journal of Mathematical Sciences. 2015. Vol. 211. No. 3. pp.310-318.

Технология поиска и сбора в Интернете текстов на малых языках России

Л.Я. Зайдельман, И.В. Крылова, Б.В. Орехов
nevmenandr@gmail.com
Москва, Национальный исследовательский университет «Высшая школа экономики»

В работе описывается созданная авторским коллективом технологическая цепочка, позволяющая обнаруживать в Интернете, проверять и скачивать тексты на малых языках России. В дальнейшем получившиеся текстовые коллекции можно использовать для построения лингвистических корпусов и разработки программных продуктов, основанных на статистических методах распределения языковых единиц в текстах. Созданная технология предполагает автоматическое обращение к поисковой машине Яндекса с определённым типом запросов, кроссвалидацию выдачи, применение чёрных списков сайтов, выкачивание страниц, освобождение текста от html-разметки и определение языка текстового абзаца. Эти действия применяются как к большому Интернету, так и к экосистеме социальной сети VK.com. Результатом работы стали текстовые коллекции на малых языках народов России, размещённые в свободном доступе в Интернете.

Ключевые слова: *информационный поиск, малые языки, лингвистические корпуса, анализ текстов на естественном языке.*

The technology of web-texts collection of Russian minor languages

Lyudmila Zaydelman, Irina Krylova, Boris Orekhov
National Research University The Higher School of Economics, Moscow, Russia

This paper describes the process that allows to detect on the web, check and download texts in minority languages of Russia. In the future, the resulting collections can be used to build a corpus and linguistic software. Technology includes an automatic query to the search engine Yandex, crossvalidation, black lists of sites, crowling, and language identification of a text paragraph. These technology applies to the Internet and to the social network VK.com. As a result, corpora of minority languages of Russia placed in free access on the Internet.

Keywords: *information retrieval, minority languages, linguistic corpora, natural language processing.*

Введение

Самым дорогостоящим этапом создания лингвистических корпусов является оцифровка печатных текстов. Существенное время и иные ресурсы уходят на сканирование, распознавание и вычитку текстов, не переведённых в электронный вид заблаговременно. Сейчас эта проблема уже неактуальна для широко распространённых языков. В случае с русским языком, например, получить большую текстовую коллекцию для создания корпуса очень легко благодаря Интернету. Сложности остаются только в той области, где у исследователей и корпусостроителей имеются специфические требования к текстам: например, в сфере исторических корпусов (нужно получить в коллекцию много текстов определённого исторического периода) или жанровых (требуются тексты определённого жанра). Но проблема создания текстовой коллекции остаётся актуальной для малых языков. Хотя история многих миноритарных языков России знает и большую периодику, и специализированные издательства, выпускающие печатную продукцию на этом языке (татарский, башкирский, удмуртский, якутский), эти тексты не оцифрованы, а, следовательно, недоступны для создания и пополнения лингвистических корпусов, а также непригодны для разработки специфического программного обеспечения, основанного на статистике распределения языковых единиц в текстах. К такого рода программному обеспечению можно отнести спеллчекеры, разного рода системы подсказок (скажем, при наборе с клавиатуры), системы определения тональности текста и многое другое.

В этой ситуации хорошим подспорьем для развития компьютерной лингистики малых языков России был бы способ находить и скачивать все тексты на данном языке, которые размещены в Интернете: на сайтах, в блогах и в социальных сетях. Последний класс текстов даже особенно ценен, так как фиксирует особую разновидность языка, не отражённую в академических грамматиках, в большинстве своём составленных ещё в советское время.

Однако такого способа до сих пор не существовало. Некоторые большие поисковые машины при определённых настройках позволяли задавать запросы так, чтобы выдача содержала в себе тексты на определённых языках, но возможность выбора этих языков была ограничена специфическим набором крупных идиомов, среди которых, разумеется, отсутствовали малые языки, в том числе и малые языки России. Поэтому мы разработали технологию, которая на данном этапе позволяет находить

тексты на малых языках России и формировать коллекции, пригодные для дальнейшего использования как в теоретических исследованиях, так и в инженерных разработках. Все полученные коллекции размещены в свободном доступе в Интернете[1].

Запросы к поисковой системе

Чтобы найти тексты на каком-либо языке в Интернете, в идельном случае следовало бы сконструировать робота, который обошёл бы все страницы в Сети, проанализировал размещённые на них тексты и определил, какие страницы содержат слова на нужном языке. Однако создание такого сервиса в современных условиях доступно только крупным технологическим компаниям. Интернет стал слишком большим и краулер без дорогостоящей инфраструктуры серверов не справится с этой задачей за разумное время. Поэтому уместнее использовать индексы, которые уже составлены большими поисковыми компаниями в процессе обхода Интернета их роботами. Конечно, мы не можем рассчитывать на прямой доступ к индексным базам поисковиков, поэтому единственный выход — это обращение к выдаче поисковой машины через программный интерфейс. В нашем случае это Яндекс.XML[2]. Этот программный интерфейс имеет ограничение в виде 1000 запросов в сутки. Показ каждой новой страницы выдачи, содержащей по 10 пунктов) считается запросом, так что на обслуживание поиска всех страниц Интернете на всех малых языках России у нас ушло около 240 суток.

Поисковый движок принимает в качестве запроса слова (а не отдельные символы и не их сочетания), так что предварительно необходимо сформировать список слов для данного языка, каждое из которых поданное в качестве запроса к поисковику, позволило бы получить выдачу из максимального числа страниц, содержащих текст на данном языке. При этом тексты на других языках должны считаться «шумом», и их число в выдаче должно быть минимизировано.

Разумным было бы составление частотного списка слов для каждого языка и их пересечение с такими же списками других языков. В этом случае наиболее частотные слова из получившихся списков, не совпадающие со словами из других списков, стали бы хорошими кандидатами для того, чтобы находить по ним тексты. Однако не следует забывать, что у нас отсутствуют текстовые коллекции, которые позволили бы предпринять такую операцию. Следовательно, подбор поисковых терминов должен производиться вручную на основе лингвистической литературы (словари и грамматики). В слова-маркеры для каждого языка должны войти служебные слова: они достаточно частотны в любом языке, но при этом подойдут не любые служебные слова, а достаточно длинные в смысле числа букв, чтобы не совпасть со служебными словами или аббревиатурами в других языках.

К сожалению, не может помочь в составлении списков слов-маркеров и Википедия, так как все языковые разделы на малых языках России в большой степени сгенерированы автоматически и естественное частотное распределение слов в этих текстовых коллекциях нарушено [1].

Составленные списки слов-маркеров отправлялись в качестве запросов к Яндекс.XML. При этом были выставлены дополнительные настройки, заставляющие поиск воспринимать запрос как точный, потому что в противном случае Яндекс стремился понять слово малого языка как опечатку в написании русского слова и исправить. Это приводило к нерелевантным результатам поиска.

Несмотря на то, что большинство слов-маркеров позволяли получить нужную выдачу, эта выдача нуждалась в перепроверке. Первый этап перепроверки: выяснить, находятся ли те же самые сайты по другому слову-маркеру для этого языка. Второй этап перепроверки: находятся ли те же самые сайты по слову-маркеру другого языка. Эти сравнения выдачи позволяли выделить как «шумные маркеры», то есть слова, которые имеются не в одном, а сразу в нескольких малых языках (частый случай для этимологически близких языков), так и «шумящие сайты», то есть страницы, на которых имеются тексты сразу на нескольких языках.

Запросы производились таким образом, чтобы обнаружить не только страницы в Интернете вообще, но и специально на сайте vk.com. В этой социальной сети мы искали сообщества, в которых имелись бы тексты на интересующих нас языках.

Полученные списки доменов распределялись по нескольким группам в зависимости от того, много или мало страниц, содержащих текст на интересующем нас языке, на этом сайте обнаруживалось. Так, понятно, что на сайте youtube.com тексты на удмуртском языке не составляют основную часть текстовой информации, хотя несколько роликов и комментариев на удмуртском языке на сайте присутствуют. Кроме того, в Сети нашлось достаточно много страниц, не содержащих полноценных текстов на малых языках России, но содержащих отдельные слова этих языков. Среди них хостинги музыки, сайты рефератов (в том числе рефератов лингвистических работ, посвящённым малым язы-

[1] http://web-corpora.net/minorlangs/

[2] https://xml.yandex.ru/

кам России) и т.д. Для таких сайтов мы формировали чёрные списки, которые позднее применяли к выдаче поисковика.

Бытовая система письма

При составлении списка слов-маркеров, которые должны стать поисковыми терминами при запросе к поисковику, нужно учитывать такую особенность, как бытовое написание слов на малых языках. Многие языки содержат в своих алфавитах знаки, отличные от русского варианта кириллицы и, следовательно, отсутствующие на стандартной раскладке русской клавиатуры.

С одной стороны, наличие в слове-маркере специальной графемы, специфичной для орфографии данного языка и отсутствующей в русском алфавите, повышает вероятность того, что по этому слову будут находиться веб-страницы с текстами именно на данном языке. С другой стороны, это понижает полноту найденного, потому что в случае с малыми языками велико число случаев, когда текст в интернете написан в бытовой системе письма (термин введён А.А. Зализняком в применении к древненовгородскому диалекту [2]), в которой не воспроизводятся специфические для алфавита графемы, и пишущий обходится только теми знаками, которые присутствуют в стандартной кириллической или латинской раскладке клавиатуры.

Отличие от бытовой системы письма, описанной Зализняком, в том, что для древненовгородского диалекта характерно наличие некоторого класса графем, каждая из которых может свободно заменяться другой, относящейся к тому же классу. Например, графемы «ять», ь, е могут заменять друг друга на письме. В случае с бытовой системой письма в национальных интернетах имеет место только односторонняя замена специфических графем на графемы русского или латинского алфавита.

Так, чувашское çук «нет, не имеется» (исторически соответствует всем известному йок в башкирском и др. тюркских языках) — хорошее слово-маркер. Но оно содержит специфическую графему, и тех текстов, где пользователи пишут «щук», мы по нему не найдем.

Определение языка

Задача автоматического определения языка, на котором написан текст, в принципе, известна и большей частью решена. Однако имеющиеся решения, в основном, основаны на случаях, когда у исследователя уже есть коллекция текстов на данном языке достаточного объёма. Поэтому в нашем случае следовало выстроить решение по определению языка иначе. Мы исходили из условия, что на каждой странице, которую мы анализируем, имеется либо текст на интересующем нас языке, либо текст, демонстрирующий переключение кодов, то есть такой, в котором какая-то часть является строкой на русском языке, а другая часть — на заранее известном нам малом языке. То есть задача сводилась к отличению русского текста (статистические характеристики его известны) от нерусского с помощью распределения буквенных триграм. Последняя часть как раз и считалась текстом на малом языке.

Заключение

В процессе выполнения проекта нами вручную составлены списки поисковых терминов (слов-маркеров для языков России), которые отправлялись в качестве запроса к Яндекс.XML. Полученная выдача проверялась таким образом, чтобы удостовериться, что сайты, на которых нашлись страницы, действительно содержат тексты на интересующих нас языках (в частности, к этим сайтам задавались дополнительные запросы с другими словами-маркерами). Позднее все обнаруженные таким образом сайты были скачены, очищены от html-разметки и размещены в свободном доступе в Интернете.

Благодаря описанной работе на данном этапе мы можем строить технологию поиска и определения языка иначе, основываясь на статистической обработке текстовых коллекций, а не на ручном труде исследователей.

Литература

[1] Орехов Б. В., Решетников К. Ю. К оценке Википедии как лингвистического источника: сравнительное исследование // Современный русский язык в интернете. — М.: Языки славянской культуры, 2014. — С. 310—321.

[2] Зализняк А. А. Древненовгородский диалект / Изд. 2-е, переработанное с учётом материала находок 1995—2003 гг. — М.: Школа «Языки русской культуры», 2004. — С. 21-23 и далее.