**Кластеризация результатов веб-поиска**

Гайдамашко Даниил Олегович,

Карпенко Максим Дмитриевич

Научный руководитель: Завриев Николай Константинович,

преподаватель ГБОУ Лицея информационных технологий №1533

# Введение

В наши дни для поиска информации крайне часто используются поисковые сайты (далее поисковики), такие как Google, Яндекс, Rambler, Bing и другие. Как правило, использование поисковика удобно и позволяет достаточно быстро найти необходимую информацию, но довольно часто для достижения результата приходится изрядно «попотеть». Главная причина этого – то, что поисковики могут выдавать в качестве результата запроса набор ссылок, описывающих разные предметы. Например, на запрос «кремень» поисковик выдаёт результаты двух основных типов: кремень-минерал и «Кремень» - телесериал. Результаты не упорядочены по смыслу, что, собственно, и является проблемой, усложняющей поиск. Конечно, можно поиск сузить, добавив в запрос дополнительные ключевые слова, однако это не решает проблему полностью, равно как и не всегда доступно. Некоторые компании, такие как Google, уже работают над решением и пытаются представить интернет в виде семантической сети, однако на данный момент это реализовано в очень малой степени.

# Постановка задачи

Одним из решений поставленной проблемы является кластеризация результатов web-поиска.

*Кластеризация результатов web-поиска* - группировка результатов поиска в поисковой системе по тому или иному признаку с целью сделать результат поиска более удобным.

Таким образом, задачей нашего дипломного проекта является создание веб-сервиса, устроенного по принципу стандартных интернет-поисковиков, который реализует кластеризацию результатов веб-поиска. Требования к ресурсу:

1. Отправлять запрос в уже существующий поисковик (Google, Yandex, Yahoo, Rambler и пр.) и получать оттуда ответы на запрос.
2. Выдавать результаты не линейно, как в изначальных поисковиках, а по группам в виде открывающихся папок. В каждой папке находится множество ссылок на веб-страницы, сходных по тематике и описывающих один и тот же объект. Над каждой папкой должен быть выведен список тегов, характеризующий этот объект, что поможет пользователю найти нужную ему папку.

# Целевая аудитория

Главной целевой аудиторией нашего проекта, безусловно, являются обычные пользователи интернета, для которых мы и стремимся упростить поиск. Также наша работа может быть интересна специалистам в области латентно-семантического анализа. Наконец, основа проекта, а именно алгоритмы кластеризации, могут быть использованы во многих информационных сервисах.

# Анализ предметной области

*Кластер* - класс родственных элементов статистической совокупности.

Существует несколько путей реализации кластеризации результатов поиска. Ниже представлены основные методы с краткой характеристикой.

1. *Custom Search Folders* – этот метод позволяет сузить результаты поиска путём распределения их по «папкам» (folders). Выбором одной из предложенных папок пользователь сужает диапазон рассматриваемых объектов. Объектами в данном случае являются HTML ссылки. Папки имеют иерархическую структуру, что дает возможность всё более и более сужать результат поиска. По сути дела папки являются центроидами кластеров, к которым затем соотносятся документы (сайты). Процесс распределения по папкам занимает не много времени, потому что матрица близости документов уже есть, она как правило считается в режиме пре-процессинга. Благодаря этому названия папок имеют читаемый вид. Таким образом, система обладает высокой скоростью работы и хорошей наглядностью. Однако папки нуждаются в периодическом обновлении своей структуры, чтобы соответствовать количеству информации в сети.
2. *Suffix Tree Clustering*. Кластеры образуются в узлах специального вида дерева – суффиксного дерева, которое строится из слов и фраз входных документов. Достоинства метода: высокая скорость работы. По времени и занимаемой памяти дерево строится пропорционально количеству документов. Наихудшая теоретическая верхняя граница времени построения - пропорционально квадрату количества документов; хорошая наглядность представления результатов. Общие фрагменты текстов и фраз выступают в качестве названия кластеров, – это имеет большой смысл, т.к. не надо затрачивать дополнительных усилий для определения подходящего имени. Недостатки метода состоят в необходимости повторной обработки текстов документов.
3. *Латентно-семантический анализ (ЛСА)* — это метод обработки информации на естественном языке, анализирующий взаимосвязь между коллекцией документов и терминами в них встречающимися, сопоставляющий некоторые факторы (тематики) всем документам и терминам.

В основе метода латентно-семантического анализа лежат принципы факторного анализа, в частности, выявление латентных связей изучаемых явлений или объектов. При классификации / кластеризации документов этот метод используется для извлечения контекстно-зависимых значений лексических единиц при помощи статистической обработки больших корпусов текстов.

LSA/LSI - это реализация основных принципов факторного анализа применительно к множеству документов. Кроме того, метод позволяет успешно преодолевать проблемы синонимии и омонимии, присущие текстовому корпусу. LSA позволяет преодолевать их, основываясь только на статистической информации о множестве документов/терминов.

# Решение

Нами был выбран метод ЛСА для использования в дипломной работе, т.к. он не нуждается в предварительной настройке на специфический набор документов, его не надо обучать, а также это лучший метод для выявления латентных зависимостей. Недостаток метода – долгая обработка запросов, содержащих сотни тысяч объектов из-за огромного количества вычислений, пресекается ограниченным количеством обрабатываемых web-страниц.

Далее нам потребуется следующая терминология:

* *Стоп слова* - слова которые встречаются в каждом тексте и не несут в себе смысловой нагрузки, это, прежде всего, все союзы, частицы, предлоги и множество других слов.
* *Стемминг* - это процесс нахождения основы слова для заданного исходного слова. Основа слова необязательно совпадает с морфологическим корнем слова. Для стемминга используется *алгоритм Портера*. Главный плюс стеммера Портера заключается в том, что он не использует никаких словарей и выделение основы осуществляется путем преобразования слова согласно определенным правилам. Недостаток алгоритма в том, что в языках есть исключения, не подходящих под правила (неправильные глаголы в английском (buy-bought), одинаково оканчивающиеся слова разных частей речи с разными лексическими значениями в русском (плоть - колоть)).

Алгоритм метода ЛСА.

1. На входе мы получаем n текстов, анализ которых далее и будем проводить.
2. Выполняется обработка текстов, которая включает в себя:
3. Удаление знаков препинания и других символов, не несущих семантического значения.
4. Стемминг слов.
5. Удаление стоп-слов.
6. Удаление чисел.
7. Удаление из текста слов, не встречающихся ни в одном из остальных текстов.
8. На основе оставшихся слов формируется частотная матрица. В этой матрице строки соответствуют индексированным словам, а столбцы — документам. В каждой ячейке матрицы указано, какое количество раз слово встречается в соответствующем тексте.
9. Выполняется сингулярное разложение частотной матрицы.
10. *Сингулярным разложением* матрицы A размером MxN называется её представление в виде:


\begin{matrix}
A=U S V ^T
\end{matrix}


где U - ортогональная матрица размером MxM, VT - ортогональная матрица размером NxN, S - матрица размером MxN, на главной диагонали которой находятся неотрицательные числа, расположенные в порядке убывания, а все внедиагональные элементы равны нолю. Диагональные элементы матрицы S называются сингулярными числами.

1. С учетом свойств матрицы S, большей частью состоящей из нулей, для получения матрицы A требуется не M столбцов матрицы U, а лишь первые min(M,N) столбцов (в примере выше - три столбца), аналогично, лишь первые min(M,N) строк матрицы VT влияют на результат произведения. Эти столбцы и строки называются левыми и правыми сингулярными векторами.
2. Достоинство сингулярного разложения состоит в том, что оно выделяет ключевые составляющие матрицы, позволяя игнорировать шумы. Согласно простым правилам произведения матриц, видно, что столбцы и строки, соответствующие меньшим сингулярным значениям, дают наименьший вклад в итоговое произведение.
3. На основе полученных в ходе сингулярного разложения матриц U и VT формируется набор вершин, имеющих свои координаты в пространстве:
4. По сути, в строках матрицы(U) содержатся координаты тега в неком многомерном пространстве, а в столбцах матрицы(VT) - координаты текста в этом же пространстве. Таким образом, множество наших вершин будет представлено множествами тегов и текстов с «привязанными» к ним координатами.
5. После формирования множества вершин осуществляется его кластеризация.

Методы кластеризации набора вершин

*Пользовательские алгоритмы*

В проекте существует несколько методов кластеризации набора вершин. Почти все методы на входе получают параметр, от которого зависит глубина кластеризации. Далее описание каждого метода в порядке, в котором они были придуманы.

**Метод 1.** Основывается на средней длине ребра остовного дерева графа, множеством вершин которого является кластеризируемый набор вершин. Сначала вычисляется средняя длина ребра, далее, если отношение длины наибольшего ребра к средней длине больше параметра, то данное ребро удаляется. Затем снова вычисляется средняя длина ребра и действия повторяются. Если отношение меньше параметра, то кластеризация завершается.

**Метод 2.** По сути, является клоном *метода 1*, с тем лишь различием, что при вычислении средней длины ребра не учитывается длина наибольшего.

**Метод 3.** Основывается на сравнении двух самых длинных рёбер. Если отношение первого по длине ко второму больше параметра, то наибольшее удаляется. Так происходит, пока отношение не станет меньше параметра.

**Метод 4.** Данный метод похож на *метод 1* и отличается тем, что средняя длина ребра вычисляется 1 раз в самом начале алгоритма и далее не пересчитывается.

**Метод 5.** Похож на *метод 4,* но вместо средней длины ребра применяется мода длин рёбер. Предпосылками к его созданию были недостаточная точность предыдущих методов и попытка автоматического вычисления коэффициента. На практике показывает большую точность, чем описанные выше методы.

У всех перечисленных методов есть два больших недостатка:

* Необходимо вводить коэффициент. Автоматически его вычислить не получается, поскольку не было замечено какой-либо зависимости его от графа.
* Необходима постобработка. После удаления длиннейших рёбер образуются кластеры, состоящие только из текстов, только из тегов или только из одной вершины, что нас не устраивает. Их необходимо объединять с ближайшими кластерами, пока все из них не придут к оптимальному виду.

**Метод 6**. Развился из постобработки в предыдущих методах. Изначально каждая вершина принимается за кластер. Кластеры сливаются с ближайшими к ним, пока каждый кластер не будет содержать минимум один тег и два текста.

*Плотностный алгоритм DBSCAN*

Алгоритм DBSCAN (Density Based Spatial Clustering of Applications with Noise), плотностный алгоритм для кластеризации пространственных данных с присутствием шума, был предложен Мартином Эстер, Гансом-Питером Кригель и коллегами в 1996 году как решение проблемы разбиения (изначально пространственных) данных на кластеры произвольной формы. Большинство алгоритмов, производящих плоское разбиение, создают кластеры по форме близкие к сферическим, так как минимизируют расстояние документов до центра кластера.

Авторы DBSCAN экспериментально показали, что их алгоритм способен распознать кластеры различной формы. Идея, положенная в основу алгоритма, заключается в том, что внутри каждого кластера наблюдается типичная плотность точек (объектов), которая заметно выше, чем плотность снаружи кластера, а также плотность в областях с шумом ниже плотности любого из кластеров. Ещё точнее, что для каждой точки кластера её соседство заданного радиуса должно содержать не менее некоторого числа точек, это число точек задаётся пороговым значением.

На данный момент в проекте DBSCAN используется как основной метод кластеризации множества вершин.

# Программная реализация

Конечным результатом нашей работы являются два продукта: исследовательское приложение и web-ресурс.

Исследовательское приложение и веб-ресурс были реализовано на языке C# в среде Microsoft Visual Studio. Использованы библиотеки численного анализа ALGLIB и библиотека стемминга Snowball. Для создания графического пользовательского интерфейса исследовательского инструмента использовались компоненты Windows Forms (части Microsoft .NET Framework 4.5.1)

Web-ресурс представляет собой приложение на основе веб-фреймворка ASP.NET MVС. Верстка веб-страниц осуществлялась с использованием языков HTML, CSS и Javascript.

Для работы с поисковой системой используется сервис Яндекс.XML, для взаимодействия с которым были найдена и доработана библиотека Yandex.XML.Search.

Используется услуга бесплатного веб-хостинга ASP.NET приложений портала Somee.com.

# Ход работы

Работа над проектом, а именно над графическим инструментом, была начата на языке C# в среде разработки Visual Studio с помощью Microsoft .NET Framework и интерфейса программирования приложений Windows Forms.

Изначальная идея заключалась в том, чтобы в одну форму приложения вводились тексты, которые бы затем обрабатывались (удалялись знаки препинания, стоп-слова, происходил стемминг оставшихся слов). После осуществления метода ЛСА мы получали бы набор вершин, кластеризировали его и визуализировали, обозначая получившиеся кластеры.

Начальный этап работы

Изначально мы поделили работу между собой на две части. Один из нас занимался лингвистической частью: стеммингом, удалением стоп-слов и т. п. Второй – «математической» частью: формированием частотной матрицы, сингулярным разложением и работой с множеством вершин.

Для реализации плана была найдена кросс-платформенная библиотека численного анализа ALGLIB, содержащая необходимый нам готовый метод сингулярного разложения матриц. Был найден отдельный список стоп-слов на английском и русском языках, который оказался несовершенным и пополнялся в течение всей работы над проектом. Из написанной в школе программы в проект импортирован класс графов, так как с набором вершин было удобнее всего работь именно как с графом. Реализованы удаление знаков пунктуации и удаление слов, встречающихся лишь в одном тексте, и формирование частотной матрицы. Далее был написан метод кластеризации на основе метода минимального остовного дерева, а также рекурсивный метод, группирующий связные вершины графа в кластеры на информационном уровне. Затем описанные выше куски алгоритма были протестированы на введённых вручную данных и совмещены.

Стемминг

Была предпринята попытка самостоятельного написания алгоритма стемминга Портера для русского и английского языков, опираясь на их теоретическое описание, однако она провалилась, и было принято решение искать готовые библиотеки стемминга.

Сравнив характеристики различных алгоритмов русского и английского стемминга (MyStem, Stemka, Snowball) мы остановились на алгоритме, разработанном с помощью Snowball - фреймворка для создания алгоритмов стемминга, и улучшенных стеммеров английского языка, а также стеммеров для некоторых других языков. Модуль на языке Python, который мы нашли (PyStemmer), включает в себя стеммеры нескольких языков, а также показывает наилучшие результаты в скорости работы.

Было принято решение реализовать интеграцию двух языков, запуская скрипт на Python, выполняющий стемминг слова, в проекте на C#. Но сделать этого не получилось из-за несовместимости языка Python и программной платформы .NET Framework. Проблема взаимодействия С# и Python решается в языке Ironpython. Поэтому было решено добавить расширение Python Tools for Visual Studio (PTVS) в исследовательское приложение, но подсоединить стеммер всё равно не удалось. Вследствие этого мы оставили идею реализации интеграции разных языков и приступить к созданию web-инструмента на Python, к которому подсоединить стеммер не составит труда. А уже значительно позже были найдены необходимые библиотеки на C#.

Разработка кластерного веб-поисковика

Для реализации веб-инструмента было принято решение использовать популярный веб-фреймворк Django для языка Python, поскольку данный язык предназначен для быстрой разработки приложений и подходит для создания веб-сайта. Кроме того, существует огромное количество готовых модулей на Python, среди которых есть и те, что могут помочь нам в решении поставленной задачи, в том числе и вышеупомянутый PyStemmer. Библиотека ALGLIB, предоставляющая нам метод сингулярного разложения матриц в исследовательском инструменте, была заменена на модуль SciPy – большую коллекцию математических алгоритмов. А на место модуля для стемминга PyStemmer был взят более расширенный Natural Language Toolkit – набор лингвистических алгоритмов.  
Уже реализованный на C# алгоритм кластеризации текстов был благополучно переписан на Python и затем обновлялся по мере открытия и изучения отдельных методов. Для разработки графического интерфейса сайта были использованы языки HTML, CSS и JavaScript.

Развитие исследовательского приложения

В то время как один из нас был занят созданием web-инструмента, второй продолжал совершенствовать исследовательское приложение. Так как стемминг подключить к C# проекту не удалось, было принято решение временно пользоваться уже отстемленными текстами в качестве тестовых данных. Работа на данном этапе сводилась к совершенствованию уже существующих методов и поиску новых методов кластеризации.

Парсинг существующих веб-поисковиков

Изначально планировалось, что созданный веб-инструмент будет получать со страницы существующего интернет-поисковика (Google, Yandex, Bing и пр.) набор ссылок на веб-страницы, с которых затем извлекал бы тексты и выполнял их кластеризацию. На основе составленных кластеров элементы поисковой выдачи бы группировались по группам, каждая из которых описывалась бы набором тегов, находящихся в одной смысловой категории.

Но на некоторых веб-страницах количество текстовой информации просто огромно, не говоря уже об уникальности структуры каждой. В результате этого получение данных сильно усложняется, а также сложнее становится фильтрация шумов и лишней информации, так что эта идея была оставлена. Тогда мы подробно рассмотрели непосредственно саму поисковую выдачу интернет-сервисов и обнаружили, что каждый результат поиска включает в себя сниппет – короткое описание веб-страницы, формируемое самим веб-поисковиком. Сниппеты формируются в каждой поисковой системе по-своему: например, поисковик Google формирует краткое описание из метатегов html-страницы, а Яндекс выполняет более сложный алгоритм, формируя сниппет из текста документа. Но, поскольку набор метатегов совершенно необязательно соответствует содержимому веб-страницы, то мы выбрали парсинг поисковой выдачи Яндекса.

Мы перепробовали много разных библиотек парсинга и краулинга веб-страниц, но остановились на фреймворке Grab. Данная библиотека позволяет извлекать данные с веб-страницы, используя регулярные выражения или XPath выражения. Благодаря Firebug Lite - расширению для браузеров Google Chrome и Mozilla Firefox – мы смогли узнать структуру страницы результатов Яндекс-поиска и подобрать подходящее XPath выражение для получения необходимых нам данных (заголовков, адресов ссылок и сниппетов).

Финальный этап работы над исследовательским приложением

После того, как были найдены библиотеки стемминга под C#, стала возможной реализация приложения в том виде, в каком оно задумывалось изначально. Таким образом, на данном этапе подключались библиотеки стемминга, улучшалась визуализация набора вершин и кластеров и удобство использования приложения.

Следующий виток в развитии

После завершения работы над пользовательскими методами кластеризации множества вершин выяснилось, что их точность и эффективность довольно низка и не соответствует нашим целям. Поэтому было принято решение в дальнейшем для кластеризации использовать алгоритм DBSCAN, отличающийся гораздо большей точностью. Код алгоритма был найден в интертете и подстроен под наш проект. Вместе с этим само приложение подверглось масштабной реконструкции. Добавился новый функционал (сохранение и открытие файлов с текстами), код был оптимизирован и реструктурирован, а результирующие данные стали гораздо понятнее для пользователя за счёт вывода полных форм слов, а не обрезанных, как раньше.

Что касается web-сервиса, то попытки интеграции двух языков, равно как и реализации парсинга поисковой страницы Яндекса в обход капчи, оказались неудачной, и мы поменяли свою стратегию. Отныне веб-инструмент разрабатывался на том же C# в Visual Studio с использованием фреймворка ASP.NET MVC, а для работы с поисковой выдачей было решено воспользоваться сервисом Яндекс.XML, предлагаемым непосредственно самой IT-компанией, принцип работы которого – возвращение поисковых данных в формате XML в ответ на запрос с предварительно зарегистрированного и добавленного в индекс сервера. Была найдена и доработана библиотека, содержащая необходимые методы и компаненты для взаимодействия с сервисом.

В конце концов web-сайт был готов, а компания Яндекс согласилась предоставить нам использование поисковой выдачи с небольшими привилегиями. Таким образом мы обрели полное и стабильное получение данных для дальнейшей обработки.

# Результат

Были изучены методы для выполнения кластеризации результатов поиска, а также разработан собственный метод кластеризации набора вершин в пространстве. На данный момент его главной проблемой является относительно фиксированная глубина кластеризации. Другой проблемой является поверхностность семантического анализа текстов.

На основе проведённых исследований было реализовано исследовательское приложение, способное производить кластеризацию нескольких текстов. Готов web-сервис, реализующий кластеризацию результатов поиска (на данный момент он находится на завершающем этапе разработки).

# Направления дальнейших разработок

Что касается разработки и развития веб-инструмента, то в первую учередь актуален вопрос улучшение работы сервиса, его интерфейса и дизайна и распространение его среди интернет-пользователей. В частности, необходимо индексирование сайта другими известными поисковыми службами. В дальнейшем возможна интеграция веб-сервиса с другими существующими поисковиками (Google, Bing и пр.).

Если говорить об алгоритмической части, то здесь мы планируем испробовать некоторые альтернативные методы кластеризации множества вершин. Помимо этого, возможны некоторые эксперименты с обработкой текстов целиком, а не только повторяющихся элементов (благо объём сниппетов это позволяет), а также попытки делать выводы о группировке результатов поиска исходя из результатов работы нескольких разных алгоритмов кластеризации. Ну и, разумеется, планируется улучшить работу уже действующих алгоритмов.

# Список литературы и используемых материалов

1. Фридл Дж. Регулярные выражения, 3-е издание. СПб.: Символ-Плюс, 2008 г.
2. Большакова Е.И. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика: учеб. пособие. М.: МИЭМ, 2011 г.
3. Edunov «Латентно-семантический анализ» [Электронный ресурс]:  
   <http://habrahabr.ru/post/110078/>
4. andreycha «Обзор алгоритмов кластеризации данных» [Электронный ресурс]:  
   <http://habrahabr.ru/post/101338/>
5. «SVD-разложение прямоугольной матрицы» [Электронный ресурс]:  
   <http://alglib.sources.ru/matrixops/general/svd.php>
6. Онлайн-книга «Изучаем ASP.NET MVC 4»  
   <http://metanit.com/sharp/mvc/>
7. «htmlbook.ru» [Электронный ресурс]:  
   <http://htmlbook.ru/>
8. «Учебник HTML» [Электронный ресурс]:  
   <http://webremeslo.ru/html/glava0.html>
9. «Учебник CSS» [Электронный ресурс]:  
   <http://webremeslo.ru/css/glava0.html>
10. «Яндекс.XML. Руководство разработчика» [Электронный ресурс]:  
    <http://tech.yandex.ru/xml/doc/dg/concepts/about-docpage/>
11. «Яндекс.Помощь. Вебмастер» [Электронный ресурс]:  
    <https://yandex.ru/support/webmaster/service/what-is-webmaster.xml>