МОСКОВСКИЙ КОМИТЕТ ОБРАЗОВАНИЯ

ЛИЦЕЙ №1533 (ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ)

**Выпускная работа**по специальности «Прикладное программирование»  
учеников группы 11.4  
Гайдамашко Даниила Олеговича,  
Карпенко Максима Дмитриевича

Кластеризация  
результатов веб-поиска

**Научный руководитель:**   
Завриев Николай Константинович  
преподаватель лицея1533 (информационных технологий)

**Москва, 2015 г.**

Оглавление

[Введение 3](#_Toc433488535)

[Постановка задачи 4](#_Toc433488536)

[Целевая аудитория 4](#_Toc433488537)

[Обзор аналогов 5](#_Toc433488538)

[Введение в предметную область 8](#_Toc433488539)

[Решение 10](#_Toc433488540)

[Программная реализация 12](#_Toc433488541)

[Ход работы 13](#_Toc433488542)

[Результат 18](#_Toc433488543)

[Направления дальнейших разработок 18](#_Toc433488544)

# 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.

# Обзор аналогов

**Полных аналогов ни графического инструмента кластеризации текстов, ни веб-инструмента кластеризации результатов интернет-поиска найдено не было, но в настоящий момент уже существуют веб-ресурсы, реализующие кластеризацию интернет-поиска, и, как следствие, упрощение поискового процесса, а также инструменты для выявления семантики текстов.**

|  |  |  |
| --- | --- | --- |
| Название инструмента | Достоинства | Недостатки |
| **Yyppi  (ранее Clusty)** | * Кластеризация предварительно отобранных результатов поиска проводится по трем критериям группировки:   - кластеризация по наиболее часто встречаемым в результатах поиска ключевым словам;  - группировка результатов поиска по поисковым системам, в которых были найдены результаты поиска;  - группировка результатов поиска по доменным зонам; * Результаты кластеризации по наиболее часто встречаемым в результатах поиска ключевым словам представляются в виде списка пунктов меню, по которым возможна пересортировка результатов выдачи; * При отображении кластеры упорядочиваются по статистике найденных слов; * Возможность выделения слов, часто встречающихся совместно со словами поискового запроса; | * Поиск с учетом словоизменения (учет морфологии) реализован только для английского языка; * Поиск по энциклопедиям реализован только на основе использования метапоиска по англоязычной части энциклопедии wkipedia, даже если запрос пользователя введен на русском языке; * Кластерный анализ не позволяет получать точные результаты поиска при вводе запроса пользователя на естественном языке; * Метапоисковая система не использует ни синтаксического, ни семантического анализа текстов; |
| **Compreno** | * Лексический, морфологический, синтаксический и семантический анализы текстов; * Использование Универсальной Семантической Иерархии (УСИ), способной описывать явления от общего к частному; * Подбор слов для перевода осуществляется из понятийного набора, который находится на ветке универсального семантического дерева и содержит в себе аналоги слова, в т.ч. и из второго языка; * Каждое слово из переводимого предложения описывается максимальным набором понятийных эквивалентов на всех уровнях смысловой иерархии, поскольку нижестоящие элементы системы по иерархии наследуют признаки вышестоящих элементов. | * Невозможность применения семантико-синтаксического анализа в массовых поисковых системах из-за очень высоких требования к компьютерным мощностям, необходимым для обработки и индексации информационных массивов на понятийном уровне. |
| **Nigma** | * Кластерный метапоиск по документам на русском. * Тщательный учет специфики русскоязычных запросов. * Мощная система исправления орфографических ошибок и опечаток в запросах. * Возможность выбирать поисковики, на которые отправляются запросы. * Альтернативный AJAX-интерфейс, в котором кластеры отображаются в виде облака тегов; * Метапоиск по базам изображений выбранных поисковиков; | * Ориентация только на один выбранный язык (русский, английский или украинский); * Приоритет – у русского языка; * Неточное описание кластеров из-за использования только одного тега, являющимся словом или словосочетанием; |

# Введение в предметную область

*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, V - ортогональная матрица размером NxN, W - матрица размером MxN, на главной диагонали которой находятся неотрицательные числа, расположенные в порядке убывания, а все внедиагональные элементы равны нолю. Диагональные элементы матрицы W называются сингулярными числами.

1. С учетом свойств матрицы W, 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 V T 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 V^t 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(V^t) - 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.

# Программная реализация

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

Исследовательское приложение было реализовано на языке C# в среде Microsoft Visual Studio. Использованы библиотеки численного анализа ALGLIB, библиотека стемминга Snowball, а так же наши школьные наработки по графам. Для создания графического пользовательского интерфейса использовались компоненты Windows Forms (части Microsoft .NET Framework 4.5.1)

Web-ресурс представляет собой приложение на основе веб-фреймворка Django, написанное на языке Python в среде разработки PyCharm. Использованы сторонние модули Scipy (коллекция библиотек численного анализа) и Natural Language Toolkit (NLTK) (пакет библиотек для символьной и статистической обработки естественного языка). Верстка веб-страниц осуществлялась с использованием языков HTML, CSS и Javascript.

# 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 показывал 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. На последних стоит остановиться поподробнее.

Методы кластеризации набора вершин

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. В данный момент этот метод основной.

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 получал 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 от 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, использую 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.

# 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 были реализованы web-сервис и исследовательское приложение, способные выполнять кластеризацию результатов поиска и нескольких текстов соответственно на приемлемом уровне на основе метода ЛСА.

# Направления дальнейших разработок

Помимо оптимизации имеющихся алгоритмов, планируется множество исправлений. Прежде всего, мы планируем в будущем в процессе ЛСА рассматривать и кластеризировать вершины текстов и вершины тегов раздельно, что объясняется понятной логикой. Также в ближайшие планы на будущее входит испытание уже существующих методов кластеризации (прежде всего dbscan). Возможно комбинированное использование нескольких методов. Ещё одно направление развития – анализ текстов целиком, а не только повторяющихся элементов, что стало возможным благодаря использованию небольших по объёму сниппетов. Это позволит терять меньше информации и в будущем полноценно использовать в проекте синонимичность слов.

Что касается развития веб-инструмента, то планируется улучшение интерфейса и дизайна сайта с использованием макетов. Также в планах находится выбор способа для более стабильного получения поисковой выдачи. Ну и конечно, главной целью является хостинг сайта для его публичного использования.