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)

Оглавление

[Введение 2](#_Toc433745903)

[Постановка задачи 3](#_Toc433745904)

[Целевая аудитория 3](#_Toc433745905)

[Обзор аналогов 4](#_Toc433745906)

[Анализ предметной области 7](#_Toc433745907)

[Решение 9](#_Toc433745908)

[Программная реализация 13](#_Toc433745909)

[Ход работы 14](#_Toc433745910)

[Результат 18](#_Toc433745911)

[Направления дальнейших разработок 20](#_Toc433745912)

[Приложения 21](#_Toc433745913)

# 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; |

# 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, 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.

Методы кластеризации набора вершин

*Алгоритм k-means (k-средних)*

Разбивает множество элементов векторного пространства на заранее известное число кластеров k. Действие алгоритма таково, что он стремится минимизировать среднеквадратичное отклонение на точках каждого кластера.

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем вершины разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров.

Не подходит для проекта, поскольку необходимо заранее знать число получаемых кластеров.

*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. В данный момент этот метод основной.

# 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.

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).

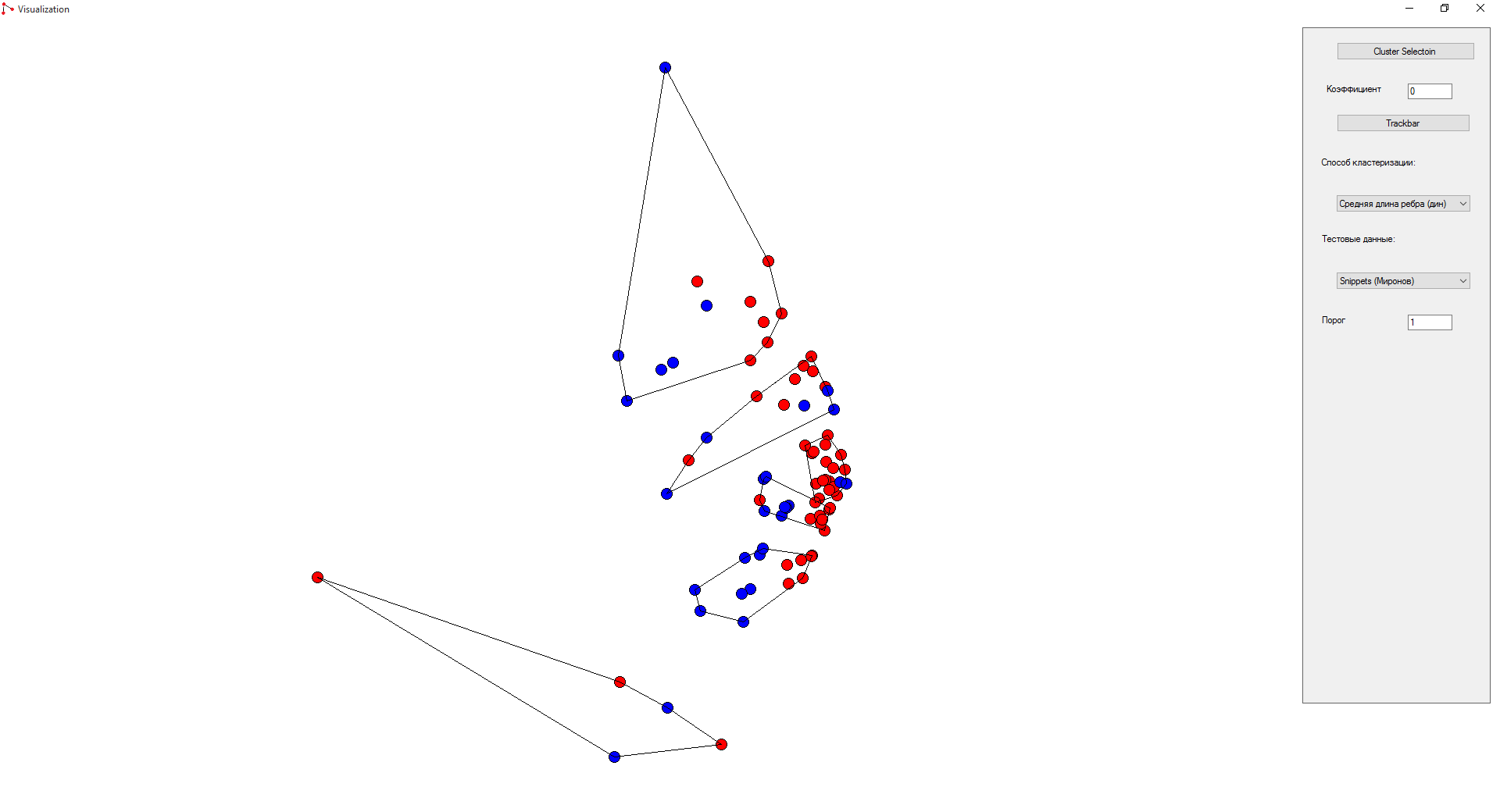
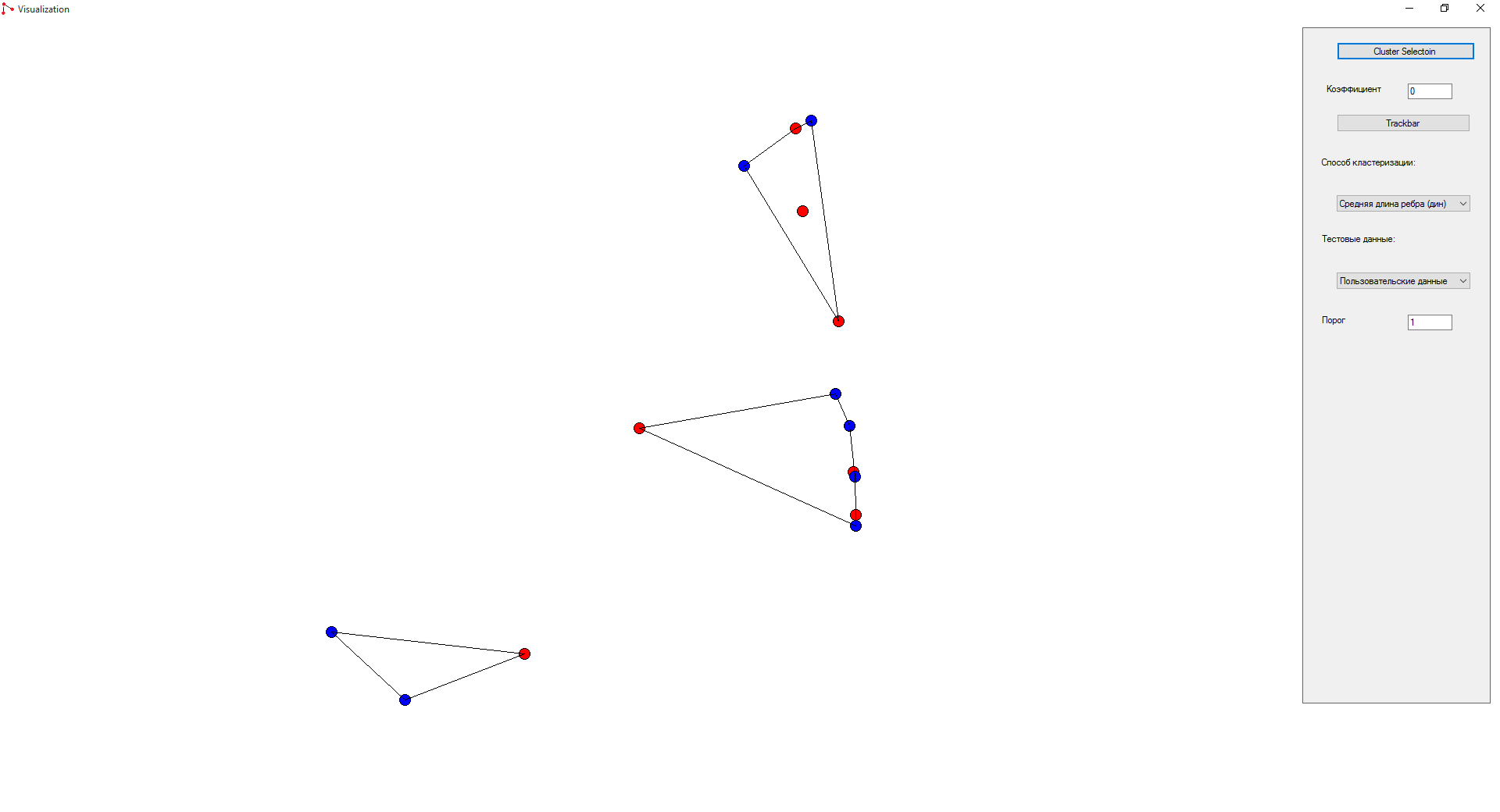
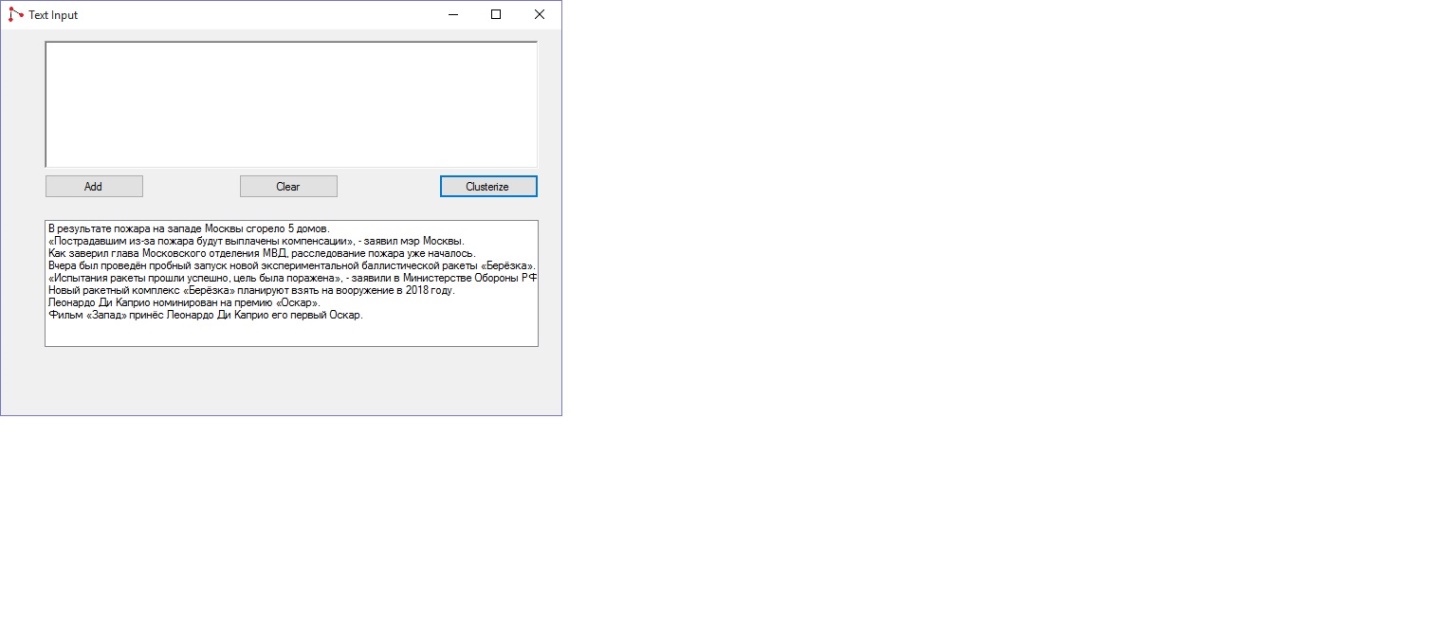
Финальный этап работы над исследовательским приложением

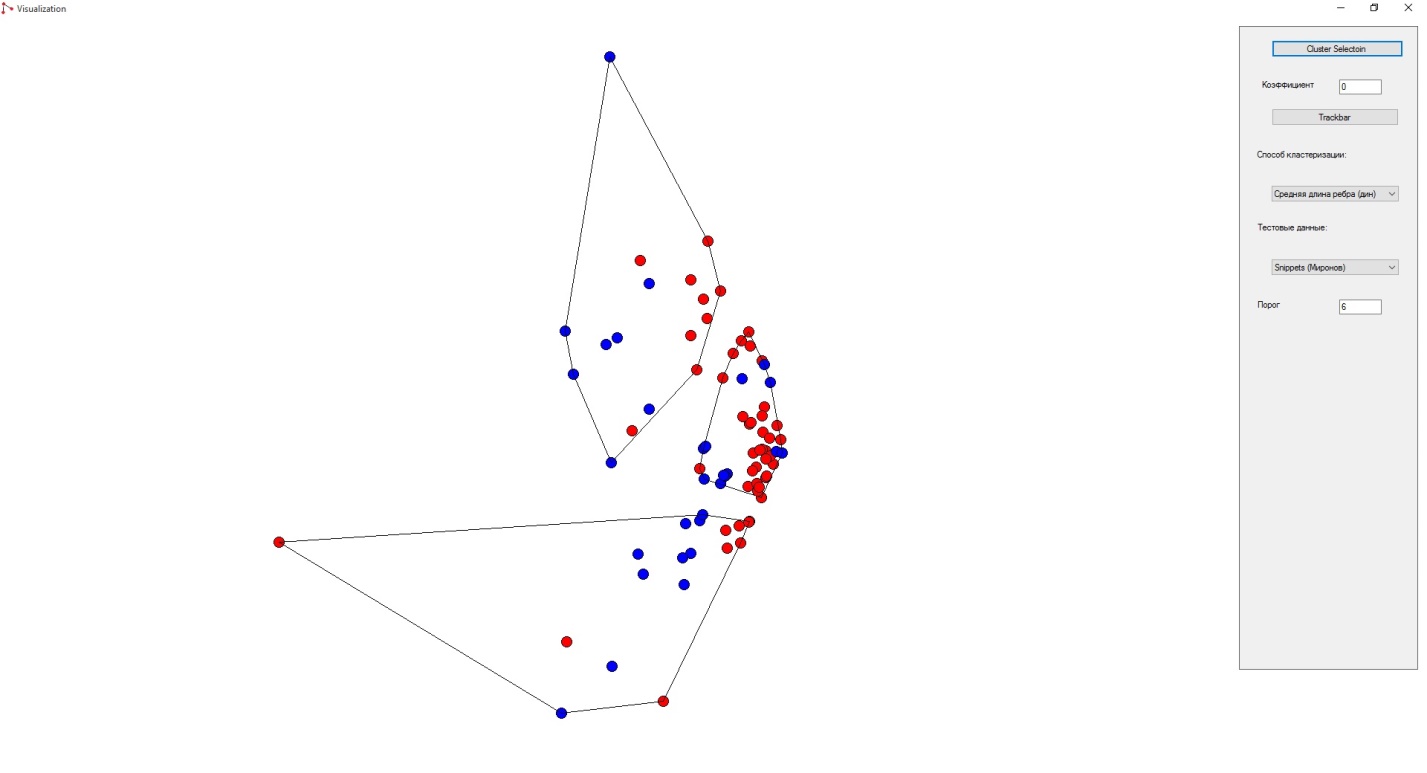
После того, как были найдены библиотеки стемминга под C#, стала возможной реализация приложения в том виде, в каком оно задумывалось изначально. Таким образом, на данном этапе подключались библиотеки стемминга, улучшалась визуализация набора вершин и кластеров и удобство использования приложения.

# 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 было реализовано исследовательское приложение, способное производить кластеризацию нескольких текстов. Почти готов web-сервис, реализующий кластеризацию результатов поиска (на данный момент он находится на завершающем этапе разработки). Что касается качества кластеризации, то получаемые нами результаты пока далеки от идеала. Только часть кластеров (60-70 %) не представляет собой смысловые группы текстов, в то время как остальные – смешанные, наполненные различными шумами. Происходит это отчасти из-за несовершенства алгоритма кластеризации, отчасти из-за низкого качества обработки текстов (несовершенство алгоритма Портера).

Скриншоты работы исследовательского приложения: 



# Направления дальнейших разработок

Помимо полной реализации web-сервиса и 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.

# Приложения

class Cluster

{

public List<Vertex> Data;

public Cluster()

{

Data = new List<Vertex>();

}

public List<Tags> GetWds()

{

List<Tags> result = new List<Tags>();

for (int i = 0; i < Data.Count; i++)

{

if (Data[i].Data is Word)

{

result.Add(Data[i].Data);

}

}

return result;

}

public List<Tags> GetTxts()

{

List<Tags> result = new List<Tags>();

for (int i = 0; i < Data.Count; i++)

{

if (Data[i].Data is TextTitle)

{

result.Add(Data[i].Data);

}

}

return result;

}

public void ConvexHull(Graphics g)

{

bool include = true;

bool first;

bool orient = true;

for (int i = 0; i < Data.Count; i++)

{

for (int j = i + 1; j < Data.Count; j++)

{

first = true;

for (int a = 0; a < Data.Count; a++)

{

if (a != i && a != j)

{

if (!LineCalculations.CheckLine(Data[i], Data[j], Data[a]))

{

if (first) { orient = LineCalculations.HighCheck(Data[i], Data[j], Data[a]); first = false; }

if (LineCalculations.HighCheck(Data[i], Data[j], Data[a]) != orient) include = false;

}

}

}

if (include) LineCalculations.ConnectDots(Data[i], Data[j], g);

include = true;

}

}

}

}

class Clusters

{

Graph G;

int longestEdgeindex;

int slongestEdgeindex;

double lEdgeWeight;

double slEdgeWeight;

List<Cluster> C;

Cluster Texts;

Tags[] textTitles;

public double r = 0;

public Clusters(double[,] A, Tags[] texts, Tags[] words)

{

textTitles = texts;

C = new List<Cluster>();

G = new Graph();

Texts = new Cluster();

double[] W = new double[A.GetLength(0)];

double[,] U = new double[A.GetLength(0), A.GetLength(1)];

double[,] VT = new double[A.GetLength(1), A.GetLength(1)];

alglib.rmatrixsvd(A, A.GetLength(0), A.GetLength(1), 1, 1, 0, out W, out U, out VT);

#region

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Singular value decomposition of a rectangular matrix.

The algorithm calculates the singular value decomposition of a matrix of

size MxN: A = U \* S \* V^T

The algorithm finds the singular values and, optionally, matrices U and V^T.

The algorithm can find both first min(M,N) columns of matrix U and rows of

matrix V^T (singular vectors), and matrices U and V^T wholly (of sizes MxM

and NxN respectively).

Take into account that the subroutine does not return matrix V but V^T.

Input parameters:

A - matrix to be decomposed.

Array whose indexes range within [0..M-1, 0..N-1].

M - number of rows in matrix A.

N - number of columns in matrix A.

UNeeded - 0, 1 or 2. See the description of the parameter U.

VTNeeded - 0, 1 or 2. See the description of the parameter VT.

AdditionalMemory -

If the parameter:

\* equals 0, the algorithm doesn’t use additional

memory (lower requirements, lower performance).

\* equals 1, the algorithm uses additional

memory of size min(M,N)\*min(M,N) of real numbers.

It often speeds up the algorithm.

\* equals 2, the algorithm uses additional

memory of size M\*min(M,N) of real numbers.

It allows to get a maximum performance.

The recommended value of the parameter is 2.

Output parameters:

W - contains singular values in descending order.

U - if UNeeded=0, U isn't changed, the left singular vectors

are not calculated.

if Uneeded=1, U contains left singular vectors (first

min(M,N) columns of matrix U). Array whose indexes range

within [0..M-1, 0..Min(M,N)-1].

if UNeeded=2, U contains matrix U wholly. Array whose

indexes range within [0..M-1, 0..M-1].

VT - if VTNeeded=0, VT isn’t changed, the right singular vectors

are not calculated.

if VTNeeded=1, VT contains right singular vectors (first

min(M,N) rows of matrix V^T). Array whose indexes range

within [0..min(M,N)-1, 0..N-1].

if VTNeeded=2, VT contains matrix V^T wholly. Array whose

indexes range within [0..N-1, 0..N-1].

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#endregion

//Добавление вершин из матриц

for (int i = 0; i < U.GetLength(0); i++)

{

G.V.Add(new Vertex(words[i], U[i, 0], U[i, 1], U[i, 2]));

}

for (int i = 0; i < VT.GetLength(1); i++)

{

G.V.Add(new Vertex(texts[i], VT[0, i], VT[1, i], VT[2, i]));

}

//добавление рёбер

for (int i = 0; i < G.V.Count; i++)

{

for (int j = i + 1; j < G.V.Count; j++)

{

G.E.Add(new Edge(G.V[i], G.V[j]));

}

}

for (int i = 0; i < texts.Length; i++)

{

for (int j = 0; j < G.V.Count; j++)

{

if (texts[i].GetTag == G.V[j].Data.GetTag)

{

Texts.Data.Add(G.V[j]);

}

}

}

}

public void ClusterSelection(double k, int n)

{

G.FindMinSpanTree(G.V[0]);

for (int i = 0; i < G.E.Count; i++)

{

if (!G.E[i].optimal)

{

G.E.Remove(G.E[i]);

i--;

}

}

LongestEdge();

// methods

#region

if (n == 1)// самое длинное/второе по длине ребро

{

while (lEdgeWeight > k \* slEdgeWeight && G.E.Count != 1)

{

G.E.Remove(G.E[longestEdgeindex]);

LongestEdge();

}

}

else

{

if (n == 2)// самое длинное/нормальная динамическая средняя длина

{

while (lEdgeWeight > k \* AverageEdgeWeight(G.E) && G.E.Count != 1)

{

G.E.Remove(G.E[longestEdgeindex]);

LongestEdge();

}

}

else

{

if (n == 3)// самое длинное/нормальная статическая средняя длина

{

double temp = AverageEdgeWeight(G.E);

while (lEdgeWeight > k \* temp && G.E.Count != 1)

{

G.E.Remove(G.E[longestEdgeindex]);

LongestEdge();

}

}

else

{

if (n == 4)// статическая мода

{

double temp = FindMode(G.E);

for (int i = 0; i < G.E.Count; i++)

{

if (G.E[i].Weight > k \* temp)

{

G.E.Remove(G.E[i]);

i--;

}

}

}

else// статическая мода (не завис. от k)

{

ModeDelete(G.E);

}

}

}

}

#endregion

Clusterize();

for (int i = 0; i < C.Count; i++)//удаление пустых кластров

{

if (C[i].Data.Count < 1)

{

C.Remove(C[i]);

i--;

}

}

for (int i = 0; i < C.Count; i++)//присоединение кластеров, сост. только из текстов

{

if (ConsistsOfTexts(C[i]))

{

AddToClosestCluster(C[i]);

i--;

}

}

for (int i = 0; i < C.Count; i++)//присоединение кластеров, сост. из слов + 1 или меньше текстов

{

if (ConsistsOfWords(C[i]))

{

AddToClosestCluster(C[i]);

i--;

}

}

}

//нахождеине самого длинного а второго по длине рёбер

public void LongestEdge()

{

if (G.E.Count > 1)

{

longestEdgeindex = 0;

lEdgeWeight = G.E[0].Weight;

for (int i = 0; i < G.E.Count; i++)

{

if (G.E[i].Weight > lEdgeWeight)

{

slEdgeWeight = lEdgeWeight;

slongestEdgeindex = longestEdgeindex;

lEdgeWeight = G.E[i].Weight;

longestEdgeindex = i;

}

}

}

}

//нахождение средней длины ребра

public double AverageEdgeWeight(List<Edge> sourse)

{

double sum = 0;

for (int i = 0; i < sourse.Count; i++)

{

sum += sourse[i].Weight;

}

return sum / sourse.Count;

}

//распределяет вершины по кластерам

public void Clusterize()

{

C = new List<Cluster>();

for (int i = 0; i < G.V.Count; i++)

{

bool exists = false;

for (int j = 0; j < C.Count; j++)

{

bool tobreak = false;

for (int k = 0; k < C[j].Data.Count; k++)

{

if (G.V[i] == C[j].Data[k])

{

exists = true;

tobreak = true;

break;

}

}

if (tobreak) break;

}

if (!exists)

{

C.Add(new Cluster());

C[C.Count - 1].Data.Add(G.V[i]);

ClusterizeStep(C[C.Count - 1], G.V[i], true);

}

}

}

void ClusterizeStep(Cluster cl, Vertex v, bool firststep)

{

if (firststep)

{

for (int i = 0; i < G.E.Count; i++)

{

if (G.E[i].V1 == v)

{

ClusterizeStep(cl, G.E[i].V2, false);

}

else if (G.E[i].V2 == v)

{

ClusterizeStep(cl, G.E[i].V1, false);

}

}

}

else

{

bool exists = false;

for (int k = 0; k < cl.Data.Count; k++)

{

if (v == cl.Data[k])

{

exists = true;

break;

}

}

if (!exists)

{

cl.Data.Add(v);

for (int i = 0; i < G.E.Count; i++)

{

if (G.E[i].V1 == v)

{

ClusterizeStep(cl, G.E[i].V2, false);

}

else if (G.E[i].V2 == v)

{

ClusterizeStep(cl, G.E[i].V1, false);

}

}

}

}

}

public bool TextsPacked()

{

bool fullpackage = true;

for (int i = 0; i < textTitles.Length; i++)

{

bool textpacked = false;

for (int j = 0; j < C.Count; j++)

{

for (int q = 0; q < C[j].Data.Count; q++)

{

if (textTitles[i].GetTag == C[j].Data[q].Data.GetTag)

{

if (C[j].Data.Count > 1)

{

textpacked = true;

}

}

}

}

if (!textpacked)

{

fullpackage = false;

break;

}

}

return fullpackage;

}

//Определяет, состоит ли кластер лишь из текстов

public bool ConsistsOfTexts(Cluster cl)

{

bool allexist = true;

for (int i = 0; i < cl.Data.Count; i++)

{

bool oneoftexts = false;

for (int j = 0; j < Texts.Data.Count; j++)

{

if (cl.Data[i].Data.GetTag == Texts.Data[j].Data.GetTag)

{

oneoftexts = true;

}

}

if (!oneoftexts)

{

allexist = false;

break;

}

}

if (allexist)

{

return true;

}

return false;

}

//Определяет, состоит ли кластер лишь из слов-тегов + 1 или меньше текстов

public bool ConsistsOfWords(Cluster cl)

{

bool ofwds = true;

int oneoftexts = 0;

for (int i = 0; i < cl.Data.Count; i++)

{

for (int j = 0; j < Texts.Data.Count; j++)

{

if (cl.Data[i].Data.GetTag == Texts.Data[j].Data.GetTag)

{

oneoftexts += 1;

}

}

}

if (oneoftexts > Form2.limit && cl.Data.Count > oneoftexts + 2)

{

ofwds = false;

}

return ofwds;

}

//Добавляет вершину к ближайшему кластеру

public void AddToClosestCluster(Vertex v)

{

Graph temp = new Graph(G.V);

for (int i = 0; i < temp.V.Count; i++)

{

for (int j = i + 1; j < temp.V.Count; j++)

{

temp.E.Add(new Edge(temp.V[i], temp.V[j]));

}

}

Vertex neighbour = G.V[0];

double shortestedgeweight = double.MaxValue;

for(int i = 0; i < temp.E.Count; i++)

{

if (temp.E[i].V1 == v)

{

if (temp.E[i].Weight < shortestedgeweight)

{

shortestedgeweight = temp.E[i].Weight;

neighbour = temp.E[i].V2;

}

}

else

{

if (temp.E[i].V2 == v)

{

if (temp.E[i].Weight < shortestedgeweight)

{

shortestedgeweight = temp.E[i].Weight;

neighbour = temp.E[i].V1;

}

}

}

}

for (int i = 0; i < C.Count; i++)

{

for (int j = 0; j < C[i].Data.Count; j++)

{

if (C[i].Data[j] == neighbour)

{

C[i].Data.Add(v);

}

}

}

}

//Добавляет кластер к ближайшему кластеру

public void AddToClosestCluster(Cluster cl)

{

Graph temp = new Graph(G.V);

for (int i = 0; i < temp.V.Count; i++)

{

for (int j = i + 1; j < temp.V.Count; j++)

{

temp.E.Add(new Edge(temp.V[i], temp.V[j]));

}

}

Vertex neighbour = G.V[0];

double shortestedgeweight = double.MaxValue;

for (int j = 0; j < cl.Data.Count; j++)//перебор вершин кластера

{

for (int i = 0; i < temp.E.Count; i++)//перебор связанных с вершиной рёбер

{

if (temp.E[i].V1 == cl.Data[j])

{

if (temp.E[i].Weight < shortestedgeweight)

{

bool access = true;

for (int q = 0; q < cl.Data.Count; q++)

{

if (temp.E[i].V2 == cl.Data[q] && q != j)

{

access = false;

}

}

if (access)

{

shortestedgeweight = temp.E[i].Weight;

neighbour = temp.E[i].V2;

}

}

}

else

{

if (temp.E[i].V2 == cl.Data[j])

{

if (temp.E[i].Weight < shortestedgeweight)

{

bool access = true;

for (int q = 0; q < cl.Data.Count; q++)

{

if (temp.E[i].V1 == cl.Data[q] && q != j)

{

access = false;

}

}

if (access)

{

shortestedgeweight = temp.E[i].Weight;

neighbour = temp.E[i].V1;

}

}

}

}

}

}

for (int i = 0; i < C.Count; i++)//перебор имеющихся кластеров

{

bool tobreak = false;

for (int j = 0; j < C[i].Data.Count; j++)//перебор вершин в кластере i

{

if (C[i].Data[j] == neighbour)

{

for (int q = 0; q < cl.Data.Count; q++)

{

C[i].Data.Add(cl.Data[q]);

}

C.Remove(cl);

tobreak = true;

break;

}

}

if (tobreak) break;

}

}

//Нахождение моды

public double FindMode(List<Edge> sourse, out double coefficient)

{

int v = 12;

double le = Double.MinValue;

double se = Double.MaxValue;

for (int i = 0; i < sourse.Count; i++)

{

if (sourse[i].Weight < se)

{

se = sourse[i].Weight;

}

if (sourse[i].Weight > le)

{

le = sourse[i].Weight;

}

}

List<List<Edge>> gaps = new List<List<Edge>>();

for (int i = 0; i < v; i++)

{

gaps.Add(new List<Edge>());

}

double av = (le - se) / v;

for (int i = 1; i < v; i++)

{

for (int j = 0; j < sourse.Count; j++)

{

if (sourse[j].Weight > i \* av + se && sourse[j].Weight <= (i + 1) \* av + se)

{

gaps[i].Add(sourse[j]);

}

}

}

int gindex = 0;

int a = gaps[gindex].Count;

for (int i = 0; i < gaps.Count; i++)

{

if (gaps[i].Count > a)

{

gindex = i;

a = gaps[i].Count;

}

}

List<Edge> temp = new List<Edge>();

for (int i = gindex + 1; i < gaps.Count; i++)

{

for (int j = 0; j < gaps[i].Count; j++)

{

temp.Add(gaps[i][j]);

}

}

coefficient = FindMode(temp);

return AverageEdgeWeight(gaps[gindex]);

}

public double FindMode(List<Edge> sourse)

{

int v = 5;

double le = Double.MinValue;

double se = Double.MaxValue;

for (int i = 0; i < sourse.Count; i++)

{

if (sourse[i].Weight < se)

{

se = sourse[i].Weight;

}

if (sourse[i].Weight > le)

{

le = sourse[i].Weight;

}

}

List<List<Edge>> gaps = new List<List<Edge>>();

for (int i = 0; i < v; i++)

{

gaps.Add(new List<Edge>());

}

double av = (le - se) / v;

for (int i = 1; i < v; i++)

{

for (int j = 0; j < sourse.Count; j++)

{

if (sourse[j].Weight > i \* av + se && sourse[j].Weight <= (i + 1) \* av + se)

{

gaps[i].Add(sourse[j]);

}

}

}

int gindex = 0;

int a = gaps[gindex].Count;

for (int i = 0; i < gaps.Count; i++)

{

if (gaps[i].Count > a)

{

gindex = i;

a = gaps[i].Count;

}

}

return AverageEdgeWeight(gaps[gindex]);

}

public void ModeDelete(List<Edge> sourse)

{

int v = 15;

double le = Double.MinValue;

double se = Double.MaxValue;

for (int i = 0; i < sourse.Count; i++)

{

if (sourse[i].Weight < se)

{

se = sourse[i].Weight;

}

if (sourse[i].Weight > le)

{

le = sourse[i].Weight;

}

}

List<List<Edge>> gaps = new List<List<Edge>>();

for (int i = 0; i < v; i++)

{

gaps.Add(new List<Edge>());

}

double av = (le - se) / v;

for (int i = 1; i < v; i++)

{

for (int j = 0; j < sourse.Count; j++)

{

if (sourse[j].Weight > i \* av + se && sourse[j].Weight <= (i + 1) \* av + se)

{

gaps[i].Add(sourse[j]);

}

}

}

int gindex = 0;

int a = gaps[gindex].Count;

for (int i = 0; i < gaps.Count; i++)

{

if (gaps[i].Count > a)

{

gindex = i;

a = gaps[i].Count;

}

}

for (int i = 0; i < sourse.Count; i++)

{

if (sourse[i].Weight > gindex \* av + se)

{

sourse.Remove(sourse[i]);

i--;

}

}

}

//Получает список списков слов в кластерах

public List<List<Tags>> GetWdsFromClst()

{

List<List<Tags>> result = new List<List<Tags>>();

for (int i = 0; i < C.Count; i++)

{

result.Add(C[i].GetWds());

}

return result;

}

//Получает список списков текстов в кластерах

public List<List<Tags>> GetTxtsFromClst()

{

List<List<Tags>> result = new List<List<Tags>>();

for (int i = 0; i < C.Count; i++)

{

result.Add(C[i].GetTxts());

}

return result;

}

public Graph GetGraph

{

get { return G; }

}

public List<Cluster> GetClusters

{

get { return C; }

}

}

public Vertex V1, V2;

public bool optimal;

public double Weight;

public Edge(Vertex V1, Vertex V2)

{

this.V1 = V1;

this.V2 = V2;

Weight = Math.Sqrt(Math.Pow((V1.x - V2.x), 2) + Math.Pow((V1.y - V2.y), 2)/\* + Math.Pow((V1.z - V2.z), 2)\*/);

}

public void Draw(Graphics g)

{

Pen p;

p = new Pen(Color.Black, 2);

g.DrawLine(p, (int)(V1.x \* Form2.GetScale) + Form1.GetForm2Width / 2 + Vertex.AX, (int)(V1.y \* Form2.GetScale) + Form1.GetForm2Height / 2 + Vertex.AY, (int)(V2.x \* Form2.GetScale) + Form1.GetForm2Width / 2 + Vertex.AX, (int)(V2.y \* Form2.GetScale) + Form1.GetForm2Height / 2 + Vertex.AY);

}

class Graph

{

float max = float.MaxValue;

public List<Vertex> V;

public List<Edge> E;

public Graph()

{

V = new List<Vertex>();

E = new List<Edge>();

}

public Graph(List<Vertex> l)

{

V = l;

E = new List<Edge>();

}

//Нахождение минимального пути между вершинами

public void FindingMinWay(Vertex first, Vertex last)

{

first.t = 0;

Vertex temp;

List<Vertex> neighbours = new List<Vertex>();

while (last.p == max)

{

temp = Mint();

temp.p = temp.t;

neighbours = new List<Vertex>();

FindingNeighbours1(neighbours, temp);

ChangeMarkerT(neighbours, temp);

}

temp = last;

while (temp != first)

{

neighbours = new List<Vertex>();

FindingNeighbours(neighbours, temp);

for (int i = 0; i < neighbours.Count; i++)

{

double w = FindEdge(temp, neighbours[i]).Weight;

if (Math.Round(temp.p-neighbours[i].p,3) == Math.Round(w,3))

{

FindEdge(temp, neighbours[i]).optimal = true;

temp = neighbours[i];

break;

}

}

}

}

//Нахождение минимального остовного дерева

public void FindMinSpanTree(Vertex first)

{

first.t = 0;

Vertex temp = first;

List<Vertex> neighbours = new List<Vertex>();

List<Vertex> Q = new List<Vertex>();

while (Gone())

{

temp = Mint();

temp.p = temp.t;

neighbours = new List<Vertex>();

FindingNeighbours1(neighbours, temp);

ChangeMarkerT2(neighbours, temp);

Q.Add(temp);

}

for (int i = 1; i < Q.Count; i++)

{

Edge e = FindEdge(Q[i], Q[i].prev);

e.optimal = true;

}

}

//

public bool IsCon()

{

for (int i = 0; i < V.Count; i++)

{

V[i].cocheck = 1;

}

V[0].cocheck = 2;

while (IsConPrior(2))

{

for (int i = 0; i < V.Count; i++)

{

if (V[i].cocheck == 2)

{

List<Vertex> n = new List<Vertex>();

V[i].cocheck = 3;

FindingNeighbours(n, V[i]);

for (int j = 0; j < n.Count; j++)

{

if(n[j].cocheck==1)

n[j].cocheck = 2;

}

}

}

}

if (IsConPrior(1)) return false;

return true;

}

//Формирование списка вершин в минимальном остовном дереве/графе оптимального пути

public List<Vertex> IsConList(Vertex begin)

{

List<Vertex> Q = new List<Vertex>();

Q.Remove(begin);

Q.Insert(0, begin);

for (int i = 0; i < V.Count; i++)

{

V[i].cocheck = 1;

}

Q[0].cocheck = 2;

while (IsConPrior(2))

{

for (int i = 0; i < V.Count; i++)

{

if (V[i].cocheck == 2)

{

List<Vertex> n = new List<Vertex>();

V[i].cocheck = 3;

Q.Add(V[i]);

FindingNeighbours(n, V[i]);

for (int j = 0; j < n.Count; j++)

{

if (n[j].cocheck == 1)

n[j].cocheck = 2;

}

}

}

}

return Q;

}

//Проверка на наличие вершины в списке

public static bool IsConVerx(List<Vertex> Q, Vertex v)

{

bool End=false;

for (int i = 0; i < Q.Count; i++)

{

if (Q[i] == v) End = true;

}

if (End) return true;

return false;

}

//

public bool IsConPrior(int n)

{

for (int i = 0; i < V.Count; i++)

{

if (V[i].cocheck == n)

return true;

}

return false;

}

//Находение ребра в списке ребер по заданным вершинам

public Edge FindEdge(Vertex v1, Vertex v2)

{

Edge temp=null;

for (int j = 0; j < E.Count; j++)

{

if ((E[j].V1 == v1 && E[j].V2 == v2) || (E[j].V2 == v1 && E[j].V1 == v2))

{

temp = E[j];

}

}

return temp;

}

//Нахождение вершины с минимальным приоритетным коэффициентом

public Vertex Mint()

{

Vertex temp=null;

for (int i = 0; i < V.Count; i++)

{

if (!V[i].wasMin)

{

temp = V[i];

break;

}

}

for (int i = 0; i < V.Count; i++)

{

if (V[i].t <= temp.t && V[i].p == max)

{

if (!V[i].wasMin)

{

temp = V[i];

}

}

}

temp.wasMin = true;

return temp;

}

//Нахождение у вершины соседей с максимальным временным приоритетом

public void FindingNeighbours1(List<Vertex> l, Vertex temp)

{

for (int j = 0; j < E.Count; j++)

{

if (E[j].V1 == temp && E[j].V2.p==max)

{

l.Add(E[j].V2);

}

else if (E[j].V2 == temp && E[j].V1.p==max)

{

l.Add(E[j].V1);

}

}

}

//Нахождение соседей вершины

public void FindingNeighbours(List<Vertex> l, Vertex temp)

{

for (int j = 0; j < E.Count; j++)

{

if (E[j].V1 == temp)

{

l.Add(E[j].V2);

}

else if (E[j].V2 == temp)

{

l.Add(E[j].V1);

}

}

}

//

public void ChangeMarkerT(List<Vertex> l, Vertex temp)

{

double w = 0;

for (int i = 0; i < l.Count; i++)

{

w = (float)FindEdge(temp, l[i]).Weight;

if((l[i].t> temp.t + w))

{

l[i].t = (float)(temp.t + w);

l[i].prev = temp;

}

}

}

//Проверка на конец прохождения по графу

public bool Gone()

{

for (int i = 0; i < V.Count; i++)

{

if (V[i].p == max) return true;

}

return false;

}

//

public void ChangeMarkerT2(List<Vertex> l, Vertex temp)

{

double w = 0;

for (int i = 0; i < l.Count; i++)

{

w = (float)FindEdge(temp, l[i]).Weight;

if (l[i].t > w)

{

l[i].t = (float)w;

l[i].prev = temp;

}

}

}

}

class LineCalculations

{

public static bool HighCheck(Vertex dot1, Vertex dot2, Vertex dot3)

{

if (dot1.GetCoordX != dot2.GetCoordX)

{

double k = (double)(dot2.GetCoordY - dot1.GetCoordY) / (double)(dot2.GetCoordX - dot1.GetCoordX);

if (dot3.GetCoordY == (dot3.GetCoordX - dot1.GetCoordX) \* k + dot1.GetCoordY) return true;

if (dot3.GetCoordY > (dot3.GetCoordX - dot1.GetCoordX) \* k + dot1.GetCoordY)

{

return true;

}

else return false;

}

else

{

if (dot3.GetCoordX >= dot1.GetCoordX) return true;

return false;

}

}

public static bool CheckLine(Vertex dot1, Vertex dot2, Vertex dot3)

{

if (dot1.GetCoordX != dot2.GetCoordX)

{

double k = (double)(dot2.GetCoordY - dot1.GetCoordY) / (double)(dot2.GetCoordX - dot1.GetCoordX);

if (dot3.GetCoordY == Math.Round((dot3.GetCoordX - dot1.GetCoordX) \* k + dot1.GetCoordY)) return true;

else return false;

}

else

{

if (dot3.GetCoordX == dot1.GetCoordX) return true;

return false;

}

}

public static void ConnectDots(Vertex dot1, Vertex dot2, Graphics g)

{

g.DrawLine(new Pen(new SolidBrush(Color.Black)), dot1.GetCoordX + dot1.GetRad, dot1.GetCoordY + dot1.GetRad, dot2.GetCoordX + dot2.GetRad, dot2.GetCoordY + dot2.GetRad);

}

}

public delegate void ParameterChangedEvents(object sender, ParameterEventArgs e);

public class ParameterEventArgs:EventArgs

{

double k;

public ParameterEventArgs(double k)

{

this.k = k;

}

public double GetPar

{

get { return k; }

}

}

public abstract class Tags

{

public abstract string GetTag

{

get;

set;

}

}

static class TestData

{

public static List<string[]> Texts = new List<string[]>();

public static void SetData()

{

Texts.Add(new string[]{ "Андре Александрович Мирон (фамил при — Мена́кер[1]; 7 март 1941, Москв — 16 август 1987, Рига) — советск актёр театр и кино, артист эстрады. Народн артист РСФСР (1980).",

"Евген Витальевич Мирон (род. 29 ноябр 1966, Саратов, СССР) — советск и российск актёр театр и кино, народн артист Росс (2004)[1], лауреат двух Государствен прем Российск Федерации.",

"Серг Михайлович Мирон (14 феврал 1953, Пушкин, Ленинград) — российск политическ и государствен деятель, депутат Государствен дум VI созыва, руководител фракц парт «Справедлив Россия» в Государствен думе, председател совет Палат депутат парт «Справедлив Россия» — член бюр президиум Центральн совет парт (2011—2013). Ран — депутат Государствен дум V созыв (2011), Председател Совет Федерац (2001—2011), депутат Законодательн собран Санкт-Петербург (1994—2001). Председател парт «Справедлив Россия» в 2006—2011 и с 27 октябр 2013 года, ран — председател Российск парт Жизни. Выставля сво кандидатур на выбор президент РФ в 2004 и 2012 год и об раз занима последн место. Председател Наблюдательн совет «Союз десантник России».",

"Руководител фракц «Справедлив Россия» в Госдум Серг Мирон предлож запрет реклам кредит и займ посредств наружн рекламы, SMS-рассылок и проч носителей. Политик подготов соответств поправк в КоАП, пишет газет «Известия». Согласн документу, за распространен реклам кредит и займ предусмотр штраф. Для физическ лиц ег величин состав до 5 тысяч рублей; для индивидуальн предпринимател — до 50 тысяч рублей, для юрлиц — до 1 миллион рублей. Банк смогут рекламирова сво услуг тольк в собствен офисах. Как говор в пояснительн записк к законопроекту, в кризис «широк реклам потребительск кредит посредств наружн рекламы, телевизион и радиовещания, SMS-рассылок, листовок, объявлен и т.д. приобрел характер массов социальн провокации». При эт «невозможн погашен населен кредит стал существен фактор не тольк экономическ проблем, но и повышен социальн напряжен в целом», отмеча автор. По мнен Миронова, запрет на реклам кредит «поможет уменьш количеств иск о невозвращен кредитах, сниз ставк по потребительск кредитам, увелич социальн ответствен гражда и укреп их финансов независимость». Он добавил, что решен гражда об обращен за кредит «должн быт хорош обдуманным, принят с должн мер ответственности, а не явля прям следств агрессивн рекламн кампаний». В комитет по финансов рынку, по слов член эт комитет коммунист Борис Кашина, поддержа законопроект пок не готовы.",

"Актер, Режиссер рост 1.82 м 8 марта, 1941 • рыб рыб Москва, СССР (Россия) дат смерт 16 авгуcта, 1987 • 46 лет Рига, СССР (Латвия) жанр комедия, драма, мелодрам супруг Екатерин Градов (развод) ... один ребенок Ларис Голубкин всег фильм 70, 1962 — 1993",

"Актер, Продюсер, Актер: Дубляж, Режиссер, Сценарист рост 1.73 м 29 ноября, 1966 • стрелец стрелец • 48 лет Татищево, Саратовск область, СССР (Россия) жанр драма, комедия, воен всег фильм 72, 1987 — 2017",

});

Texts.Add(new string[]{"Британск полиц знает о местонахожден основател WikiLeaks",

"В суд США начина процесс прот россиянина, рассыла спам",

"Церемон вручен Нобелевск прем мир бойкотир 19 стран",

"В Великобритан арестова основател сайт Wikileaks Джулиа Ассандж",

"Украин игнорир церемон вручен Нобелевск прем",

"Шведск суд отказа рассматрива апелляц основател Wikileaks",

"НАТО и США разработа план оборон стран Балт прот Росс",

"Полиц Великобритан нашл основател WikiLeaks, но, не арестова",

"В Стокгольм и Осло сегодн состо вручен Нобелевск прем",

});

Texts.Add(new string[]{"Кремен (мини-сериал) Кремен (Kremen) Год 2012 (1 сезон) Стран Росс слога «Его правд - ег оружие» режиссер Владимир Епифанцев, Александр Аншютц сценар Миха Шульма продюсер Алекс Моисеев, Давид Дишдишян, Светла Слитюк, ... оператор Виктор Гончар композитор Игор Баба художник Евген Драбкин, Жан Сердюк жанр боевик, криминал, детект",

"Кремен (праслав. \*kremy, род. п. kremene; ст. слав. кремы, род. п. кремене)[1] — минеральн образование, состоя из кристаллическ и аморфн кремнезём (SiO2) в осадочн горн породах. Част окраш окисл желез и марганц в разн цвета, с плавн переход межд ними.",

"Росс Детективы, Боевики, Русск Режиссер Владимир Епифанцев, Александр Аншютц В рол Владимир Епифанцев, Павел Климов, Анастас Веденская, Серг Векслер, Иван Краско, ещ Отечествен кинематографист замахнул на святое, реш снят российск аналог кинохит всех врем и народ «Рэмбо: перв кровь». Так на экра вышел мини-сериа под назван «Кремень» с Владимир Епифанцев в главн роли. В результат получ криминальн боевик про оборотн в погон и отчая борьб с ними. Посмотрет на то, как лих русск Рэмб справля с продажн русск копами, можн онлайн. Слога фильм – «Его правд – ег оружие».",

"минер. горн порода, разновидн кремнезёма, очен твёрды минерал, состоя из кварц и халцедон (конкрец SiO2 в осадочн горн породах) ◆ И опя ползём мы по щел вверх; отсюд уж заметн горизонтальн тонк сло кремня, параллельн друг друг скрепля мелов породу. В. И. Немирович-Данченко, «Свят горы», 1880 г. (цитат из Национальн корпус русск языка, см. Список литературы) ◆ Зде пласт лежат плитняк, глина, мелк кремен и чёрна галька, а мест ест такж прослойк и друг землист пород, то тёмные, то иссера-жёлтые, то совс беловатые. Н. С. Лесков, «Гора», 1888 г. (цитат из Национальн корпус русск языка, см. Список литературы) ◆ Кремен — главн материа для топоров, стрел, нож и проч нужн инструментов, — а такж издел из нег использова для обм на медные,"});

Texts.Add(new string[]{"Кандидат в президент США: Обам не смог постав Путин на колени С точк зрен Рик Перри, представител Республиканск партии, Обам долж был постав Украин вооружение, а такж заключ больш количеств контракт по поставк газ в Европу. Об эт он заяв в эфир Fox News.",

"Бара́к Хуссе́йн Оба́ма-младш (англ. Barack Hussein Obama II, произнос [bəˈrɑːk huːˈseɪn oʊˈbɑːmə]; род. 4 август 1961, Гонолулу, Гавайи, США)[2] — действ (с 20 январ 2009 года) 44-й президент Соединён Штат Америки. Лауреат Нобелевск прем мир 2009 года. До избран президент был федеральн сенатор от штат Иллинойс. Был переизбра на втор срок в 2012 году. Перв афроамериканец, выдвинут на пост президент США от одн из двух крупн партий[3], и перв в национальн истор глав государств темнокож президент, а такж президент с фамил африканск и средн имен арабск этимологическ происхождения. Обам — мулат, но, в отлич от большинств чёрных американцев, не потомок рабов, а сын студент из Кен и бел американк (Стэнл Энн Данхэм). Выпускник Колумбийск университет и Школ прав Гарвардск университета, где он такж был перв за всю ег истор афроамериканцем-редактор университетск издан «Harvard Law Review». Обам такж работа обществен организатор и адвокат в област гражданск прав. Преподава конституцион прав в Чикагск институт юридическ наук с 1992 по 2004 год и одновремен трижды, в период с 1997 по 2004 год, избира в сенат штат Иллинойс. Посл неудачн попытк баллотирова в 2000 год в Палат представител США в январ 2003 год баллотирова в Сенат США. Посл побед на праймериз (первичн выборах) в март 2004 год Обам произнёс основн реч на Демократическ национальн съезд в июл 2004 года. Был избра в Сенат в ноябр 2004 года, набра 70 % голосов. Как член Демократическ меньшинств в Конгресс 109-го созыва, он помог созда закон о регулирован обычн вооружен и увеличен прозрачн в использован государствен бюджета. Он такж соверш официальн поездк в Восточн Европ (в том числ в Россию), на Ближн Восток и в Африку. Во врем работ в Конгресс 110-го созыв участвова в создан законов, каса мошенничеств на выборах, лоббизма, изменен климата, ядерн терроризм и демобилизова американск военнослужащих. Обам объяв о своём желан баллотирова в президент в феврал 2007 год и в 2008 год на президентск праймериз на Демократическ национальн съезд был официальн выдвинут от Демократическ парт кандидат в президент вмест с кандидат на пост вице-президент — сенатор от штат Делавэр Джозеф Байденом. На президентск выбор 2008 год Обам оперед кандидат от прав Республиканск парт Джон Маккейна, набра 52,9% голос избирател и 365 голос в коллег выборщик прот 45.7% и 173 у Маккейна. 9 октябр 2009 год получ Нобелевск прем мир с формулировк «за экстраординарн усил в укреплен международн дипломат и сотрудничеств межд людьми»[4].[⇨] На президентск выбор 2012 год Обам оперед кандидат от Республиканск парт Митт Ромни, набра 51,1% голос избирател и 332 голос в коллег выборщик прот 47,2% и 206 у Ромни.",

"Обам признался, что политик санкц не работа В Америк дискусс об ужесточен антироссийск санкц идут в преддвер выбор нов президента. А что там? Может, посл Обам будет попроще? Две отличительн черт эт кампан уж очевидны. Во-первых, эт характерн династийность, пот что опя Буш (тепер Джеб) и Клинтон (тепер Хиллари). Выигра что те, что другие. Москв примерн понятно, чег ждать, — переосмыслен подход к Росс будет, скорее, ещ через президентство. Но будет тепер и ещ одна, совершен нов черт предвыборн кампан в США. Еще недавн главным меньшинств в США был афроамериканцы, тепер их по числен обошл избирател латиноамериканск кровей. Сам именит кандидатом-республиканц явля Джеб Буш — сын президент Буша-старш и брат президент Буша-младшего. У нег с латиноамериканц все прост замечательно. Супруг — мексиканка, и в сем он говор на испанском. Но и Хиллар Клинтон тепер ест что продать избирателям-латиноамериканцам, а имен решен е однопартийц Обам пойт на нормализац отношен с Кубой. Уже объявл об обмен посольствами. А тепер вспомн подвешенные бюллетен в подвешенном штат Флорид на президентск выбор 2000 года. Это был бесконечн пересчеты. Губернатор Флорид там был тогд Джеб Буш. Побед ег брат Джордж. Минимальн перевес обеспеч в том числ кубино-американск избиратели. Они традицион — за республиканцев. Каза бы, Буш неч беспокоиться. Как, мол, там ни мечут демократы, уж кубино-американск избирател за пазухой. Но вот ещ две нов черты. Во-первых, из числ кандидатов-республиканц кубино-американц куд ближ не Джеб Буш с ег женой-мексиканкой, а два так подпира ег кандидата, как сенатор Крус и особен сенатор Рубио, а он об — этническ кубинцы. Во-вторых, с памятн 2000 год Флорид наводн уж не тольк кубинц из железобетонно-антикоммунистическ волны, котор бежа туд от революц и традицион голосова за республиканцев, — все больш стал иммигрант свежих, уеха с Куб част не из-з идеологии, а в поисках, например, работы. Так нов кубинцы-иммигрант только-тольк высидели гражданств США, стал избирател и част жела как раз примирен с Гаван — им так прощ езд к родственникам. То ест баланс сдвинулся. Демократ продолжа ег сдвига ещ дальше. Они намекнул на нормализац отношен тепер и с Венесуэлой. А эт уж совс по наш душу, пот что имен Куб и Венесуэл — сам близк Росс латиноамериканск республики. И что будет, есл США их от нас уведут? Впрочем, уведут ли? И вот тут пригод сраз нескольк совпадений. Во-первых, недавн вышл нов книг о Куб легендарн советск разведчик Никол Леонова. А, во-вторых, 4 июл - Ден независим США, а 5-го — Ден независим Венесуэлы. Но из-з того, что в эт год об эт дат пришл на выходные, американск и венесуэльск посольств уж провел праздничн приемы, а дипломатическ раут — эт ещ как лакмусов бумажка. Так кто ког куд завлека и как развлекал? Сам необычн участник венесуэльск прием — юнош и девушки, у котор на плеч — нашивк со сво флагом, но на погон — русск букв К. Это курсанты, котор учат в воен вуз стран ОДКБ. Отношен Москв и Каракас стал сейчас так масштабными, что для венесуэльск прием аренд огромн зал в отел Метрополь. Сред гост мног не тольк дипломатов, но и представител российск бизнеса, прич реальн сектора. Кстати, российск триколор сопровожда венесуэльск независим с сам начала. Когд наш гер Франсиск де Миранд приб ко двор Екатерин Великой, уж тогд обсужда наш независим от испанск короны. А пот наш освободител Симон Боливар принес мног народ Латинск Америк независимость, свободу, отказ от тиран и подавления. И мы очен горды, что в Росс отмеча 204-ю годовщин наш независимости, — заяв Хуа Висент Паредес, посол Венесуэл в России. На 35 лет раньш имен Росс и все та же Екатерин Велик перв в мир призна и независим США. Поэт в Москве, как нигде, символичн то, что на дипприем в чест годовщин независим США посл вынос их флаг звуч гимн страны-хозяйк России. На стен резиденц США в Москв — фотограф о сам разн этап наш двусторон отношений. На прием нынешн посол Джон Теффт повел себ крайн сдержанно: в реч он лиш раз упомянул расхождения, а при посвят культуре. Особ популярн пользова постер с Фрэнк Синатрой: в эт год — 100 лет со дня ег рождения. По мо ощущениям, культурн обм растет. Конечно, над посмотрет точн цифры, но явн не снижается. Мы провод мног музыкальн концертов. Сегодн у нас — групп Сем Джонс. До эт он спел в разн город Росс и выступ в Москве. А недавн мы организова совместн концерт Игор Бутма и Теренс Бланшара, котор счита в США одн из лучш трубачей. Роскошн джаз, — призна Теффт. Национальн культур был и у венесуэльцев. Сам трогательн момент — вручен подарк 7-летн Саш Каплунову, победител уж 4-го конкурс рисунк Венесуэл глаз русск детей. Совс друг поколен чествова ран в Госдуме. Там прошл презентац нов книг легендарн советск разведчика-латиноамериканист Никол Сергеевич Леонов о Раул Кастро. - Никола Сергеевич, поч вдруг Раул Кастро? Времен мног прошло, врод всем все известно? - Времен прошл действительн много. С момент мо знакомств с Раул Кастр прошл 62 года. В книг — и их перв совместн фотограф — тогд он ещ был студент — и масс удивительн детск фотограф сам Рауля, и, естественно, Раул с брат Фидел и с советск товарищ по оружию. Межд тем, ровн в тот ден был объявлено, что у США тепер будет в Гаван посольство, как у Куб в Вашингтоне. Почему? Наш президент реш взят курс на нормализац отношен с Кубой. Он очен хочет, как мне кажется, налад связ с кубинц в новом, расширен ключе. Он четк сказал, что стар политик не работа и мы попробу вест другую, — отмет Джон Теффт. Но вед стар политика — эт политик санкций. Значит, он не работает. Куб их выдержа благодар поддержк Москвы. А друг позна в беде. Так друз остаются. Как бы стран ни был больш друг Кубы, так стран всегд будет испытыва больш выгод от стратегическ преимуществ Кубы. Оттуд все можн делать. И над прост радоваться, что у нас с не так хорош и добр отношения, — счита Никола Леонов. Но верн и то, что нормализац отношен с кубинц — то немногое, что Обам может приписа себ как внешнеполитическ успех. В Вашингтон уж идут дальше. Тепер нормализац обеща и Венесуэле, прот котор был немедлен введ санкции, сто то стран объяв о независим курсе. И опя же кто помог и кто сказал, что, конечн же, непрост внутрен процесс — эт дел сам венесуэльцев? Со времен перв встреч командант Чавес с президент Путин Росс стал для нас не прост ещ одн стран — эт братск страна. Так говор командант Чавес. Его дел продолжа президент Николас Мадуро. Это сближен служ дел создан многополярн мира. В эт — практик России. А из эт проистека суверенитет и цельност границ. Вот что мы ищ — мира, котор мы нес и друг народам, — сказа Хуа Висент Паредес. Не буд сравнива прием по случа Дне независим Венесуэл и США, замет только, что ни в посольств США не был кубинц с венесуэльцами, ни у венесуэльц не был американцев. Межд тем в российск Интернет гуляет занятн викторина. Поч в 1963 год Фидел Кастр в Москв нос сраз пар часов? В числ проч предлага верс тогдашних, врем холодн войны, конечно, западн пропагандистов. Мол, Фидел Кастр на сам дел тайн пита страст к роскоши. Или что Фидел разбогател на доверчив советск лидеров. На сам дел загадк проста: одн час на рук Фидел показыва местное, московское, время, а втор — домашнее, гаванское."});

Texts.Add(new string[]{"Судьба — совокупн всех событ и обстоятельств, котор предопредел и в перв очеред влия на быт человека, народ и т. п.; предопределён событий, поступков; рок, фатум, доля; высш сила, котор может мысл в вид природ ил божества; древн грек персонифицирова судьб в виде: Мойр (Клото, Лахезис, Атропос), Тиха, Ате, Адрастеи, Хеймармене, Ананке; древн римлян — в вид Парк (Нона, Децима, Морта); слово, част встреча в биографическ текстах.",

"Люд всех культур во все врем сталкива с эт проблемой, с эт двум подход к жизни: все ли предопредел ил мы мож каким-т способ измен ход событ — сил наш воли, наш желаний, наш слез, преодолен нам опасн и исправлен ошибок? Все известн нам культур и народ задава так вопросом, с ним был связа даж определен божества. Например, о традиц и истор Древн Рим нам известн сегодн (а сегодн нам известн горазд больше, чем деся лет том назад, когд считалось, что Рим основа тольк Ромул и Рем, когд не придава должн значен произведен Вергил и не учитыва символическ смысл миф об Эне и маленьк групп ег соратников), что римлян вер в Бог (он упомина в мифах), стоя выш Юпитера, божеств без имени, котор называ прост — Неизвестный.",

"Судьб – многозначн термин, обозначающий: 1. предназначен (идеал); а) небесная: полн раскрыт Образ и Подоб Божия, жизн в Царств Небесном; б) земная: исполнен земн предназначения; земн реализац дан от Бог сил во слав Божию; 2. жизнен путь: исполнен ил неисполнен предназначен (например, Саул не исполн его, а Давид исполнил); 3. стечен обстоятельств (что ест Промысл Божий); 4. рок (неотвратимость)",

"Сериа Судьба: Ноч схватки. Клинк бесконечн край/Fate-stay Night: Unlimited Blade Works 2 сезон онлайн Как и в прежн времена, в город Фуюк продолжа идт магическ битв за Свят Грааль. К ней, в сво очередь, присоединя ученик школ Сир Эмия, котор оказа в неизведа для себ мире. Все происход быстр и без четк позиции, альянс распада быстрей, чем создаются, а предател окружа повсюду. И в эт противостоян героев, в сюжет раскрыва нов история, котор покажет нам кое-чт про происхожден главн героев, а так же их отношен межд собой. Наш реальност состо из множеств пут и перекрестков, вступ на один из них, мы мож полност измен сво представлен о реальности, о сюжете, как здесь, когд оказыва под багрян светил на пустыр из клинков. У «Ноч схватки» тепер нов девиз, котор звуч так «Побольш дел, поменьш слов». В фильм представл отличн анимация, на сам современ уровне. Теперь, сражен в мир «Type-Moon» будут ещ восхитительней. Оригинал: Fate-stay Night: Unlimited Blade Works Жанр: аниме, приключения, фэнтез Страна: Япон Вышел: 2015 Режиссер: Миур Такахир",

"Сериа Судьба: Ноч схватки/Fate-stay Night онлайн Потеря сво приемн отца, молод парен Сир Эми жил сам в огромн поместье. К 16-ти год парен вырос добрым, трудолюбив и хозяйственным, поэт ег окружа забот и вниман сраз две девушк — младш школьн подруг Сакур Мат и учительниц Тайг Фудзимура, котор формальн был ем опекуном, а на дел — скор старш сестрой. Всё пал прах посл того, как Сир узна о том, что ег родн городишк Фуюк явля арен магическ сражен за Свят Грааль, котор провод раз в нескольк поколений. Вот тольк жизн станов всё быстрее, и хот с последн битв прошл всег 10 лет — уж грядет нов война. Древн закон гласят, что в «королевск битве» участв сем Мастеров-магов, котор вызыва себ по одн Слуг — вечн гер прошедш ил будущ времени. Мастер, оста в живых, станов обладател Граал — величайш сокровища, исполня люб желание. В величайш битв нет никак правил, друг может стат соперник и предать, без промедлен нанест удар в спину. Узнавш горьк правд Сир долж соверш для себ тяжел выбор — отступить, отказа от борьб ил пойт в бой, риску жизн и постав на карт все, что у нег ест — невелик (как он сам считает) магическ способности, горяч сердце, безудержн желан спаст всех тех, кто для нег дорог. И когд все был уж почт решено, последн капл на чаш вес доблест и чест окажут зелен глаз из далек времен, засия в ту сам ночь Оригинал: Fate-stay Night Жанр: драмы, приключения, аним",

"Судьба — совокупн всех событ и обстоятельств, котор предопредел и в перв очеред влия на быт человека, народ и т. п.; предопределён событий, поступков"

});

Texts.Add(new string[]{"Андре Александрович Мирон фамил при рожден - Менакер 7 март 1941 Москв - 16 август 1987 Риг - советск актёр театр и кин артист эстрад Народн артист РСФСР 1980",

" Мирон - русск фамил образова от мужск имен Известн носител Мирон Александр Александрович род 1961 - советск и российск актёр Мирон Александр Васильевич 1902-1980 - советск архитектор",

" Сайт содерж больш объ информац связа с деятельн народн артист Описан фильм спектакл реценз интерв",

"Обзор политическ деятельн обществен инициат Комментар к событ в обществен жизн Росс Материал интернет- и пресс-конференц Интернет-приёмн",

"Поиск и просмотр изображен в интернет",

"Мирон Серг Депутат Государствен дум глав фракц Справедлив Росс в Госдум пят созыв",

"18 август INTERFAX.RU - Замоскворецк суд Москв приговор активист Олег Миронов к наказан в вид трех лет лишен свобод в колон строг режим",

"Посл окончан в 1982 год восьм класс Евген Мирон поступ в Саратовск театральн училищ им И А Слонов на курс Валентин Александровн Ермаков",

"Рост 173 м Дат рожден 29 ноябр 1966 48 лет Мест рожден Татищев Саратовск област СССР Росс Лучш фильм Птица-Гогол В август 44-го На Верхн Масловк Любов Перед рассвет актер",

"Детств Евген Миронов Мирон Евген Витальевич род в город Сарат 29 ноябр 1966 год",

"Биограф истор жизн Миронов Андре Александрович Подарок для женщин Андре Мирон род 07 08 по документ март 1941 в актерск сем",

"Борис Мирон 25 апрел 2013 год в грандиозн теле-шо Прям лин с Владимир Путин котор смотрел и слуша миллион",

"Мирон Менакер Андре Александрович Родител Мар Владимировн Миронов и Александр Семёнович Менакер",

"Об эт Газете.Ru сообщ адвокат Иван Мирон котор защища друг фигурант эт дел Митяев",

"Мирон Миха Евгеньевич Род 17 ноябр 1981 год в Новосибирск Оконч Сибирск государствен университет пут сообщен 2003",

" Род 7 март 1941 год в Москв в сем Похорон Андре Мирон 20 август 1987 год в Москв на Ваганьковск кладбищ участок № 40",//Andrey Mironov

"Борис Сергеевич Мирон род 29 август 1951 год в город Могоч Читинск област Служ на границ",

"О распространен фамил Мирон можн суд по количеств абонент телефон номер ил владельц телефон В Москв например таков 7969",

"Биограф Работ в театр кин на телевиден и рад на эстрад Перечен книг и стат об актер Песн в формат mp3 текст пес",

"Евген Мирон род 29 ноябр 1966 год в Саратов в сем рабоч В 1986 год оконч обучен Евген Мирон отправ в Москв где доб встреч",

" Серг Мирон очен скор ве украинск народ един фронт выступ прот временщик Порошенк",//"www.mironov.ru"

"Андре Мирон появ на свет в Москв 7 8 март 1941 год в сем артист Мар Миронов и Александр Менакер",

"Андре Мирон род 7 март 1941 год в Москв но родител в документ записа ден рожден сын на Международн женск ден",

"На презентац книг Борис Миронов Ура-путинизм Кто толка Росс к гражданск войн Борис Мирон Евген Копыш",

"Мирон Андре Александрович Народн артист РСФСР 1980 Лучш актер год по опрос журна Советск экра 1987",

" 706 Оценк 69539 Год событ 1995 Мирон В Аз бях на таз войн 892 k Оценк 9.00\*109 Чечн Translation",//"Mironov V Assault on Grozny Downtown"

"По окончан училищ был приглаш на работ в Саратовск ТЮЗ однак Евген Мирон реш продолж обучен актерск мастерств в Москв поступ в",

"sergey\_mironov Запис Календар О дневник Пред Вест от Серге Миронов",//"www.mironov.ru"

"Серг Михайлович Мирон род 14 феврал 1953 год в город Пушкин Ленинградск област Отец Миха Емельянович Мирон",

"Выступлен Борис Миронов на международн конференц Представител прав организац Итал Грец Англи в Москв"

});

Texts.Add(new string[] { "Кремен — минеральн образован состоя из кристаллическ и аморфн кремнезём SiO2 в осадочн горн пород Част окраш окисл желез и марганц в разн цвет с плавн переход межд ним",

"Рейтинг: 6.6/10 Боевик кримина детект Режиссер Владимир Епифанц Александр Аншютц В рол Владимир Епифанц Павел Клим Анастас Веденск и др",

"Оригинальн назван Кремен Стран Росс Жанр боевик детект кримина Качеств SATRip ... - Кадр из сериа Кремен Сериа Кремен смотрет онлайн",

"Перв не напада но пробуд в нем звер почувств сил человек котор называ Кремен ... смотрет онлайн Кремен (2012) в хорош качеств",

"Сериа Кремен рассказыва интересн истор о жизн человек для котор жизн друг люд наход на позиц выш чем сво собствен",

"Экшн боевик Майор Шаман бывш участник крапов берет поеха в маленьк российск городок в гост к сво стар боев товарищ Но как тольк он туд приезжа",

"Кремнев щебен и др цветн декоративн камен крошк опт розниц",

"КРЕМЕНЬ — м сам тверд и жестк из прост камн служ прежд особ для добыч огн до самогарн спичек о человек крепк нрав тверд стойк безжалостн скуп",

"кре-ме?н 1 минер горн пород разновидн кремнезём очен твёрды минера состоя из кварц и халцедон конкрец SiO2 в осадочн горн пород 2 част зажигательн устройств высека искр"

});

}

}

static class TextOperations

{

static StWdsEng Eng;

static StWdsRus Rus;

static EnglishStemmer EngStem;

static RussianStemmer RusStem;

public static List<string> ts;

public static List<List<Word>> Tag;

static double[,] Matrix;

static Tags[] TextTitles;

static Tags[] Words;

static List<string> AllWds;

public static void InitParams(string[] t)

{

string[] temp = t;

Eng = new StWdsEng();

Rus = new StWdsRus();

EngStem = new EnglishStemmer();

RusStem = new RussianStemmer();

ts = new List<string>();

ts.AddRange(temp);

Tag = new List<List<Word>>();

}

public static void InitParams(List<string> t)

{

Eng = new StWdsEng();

Rus = new StWdsRus();

EngStem = new EnglishStemmer();

RusStem = new RussianStemmer();

ts = new List<string>();

ts = t;

Tag = new List<List<Word>>();

}

public static void TagNullifier()

{

Tag = new List<List<Word>>();

}

//Дробление текста на слова

public static List<string> Divid(string txt)

{

List<string> words = new List<string>();

string[] temp = txt.Split((string[])null, StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < temp.Length; i++)

{

if (alphabetCharsExistance(temp[i]))

{

words.Add(PunctuationDel(temp[i]));

}

}

return words;

}

//Проверка на наличие в единице текста букв

static bool alphabetCharsExistance(string word)

{

for (int i = 0; i < word.Length; i++)

{

if (word[i] == 'Ё' || word[i] == 'ё') return true;

if (((int)(word[i]) >= 1040 && (int)(word[i]) <= 1071) || ((int)(word[i]) >= 65 && (int)(word[i]) <= 90)) return true;

if ((((int)(word[i]) >= 1072 && (int)(word[i]) <= 1103)) || ((int)(word[i]) >= 97 && (int)(word[i]) <= 122)) return true;

}

return false;

}

//Стемминг слов в тексте

public static List<string> vClusterize(string text)

{

List<string> tx = Divid(text);

for (int i = 0; i < tx.Count; i++)

{

string temp;

int lang = LanguageCheck(tx[i], out temp);

switch (lang)

{

case 1://Русский язык

{

tx[i] = temp;

tx[i] = RusStem.Stem(tx[i]);

if (Rus.Check(tx[i]))

{

tx.RemoveAt(i);

i--;

}

break;

}

case 2://Английский язык

{

tx[i] = temp;

if (Eng.Check(tx[i]))

{

tx.RemoveAt(i);

i--;

}

else tx[i] = EngStem.Stem(tx[i]);

break;

}

}

}

return tx;

}

//Наличие символа в массиве

static bool SymbExistance(char[] arr, char c)

{

for (int i = 0; i < arr.Length; i++)

{

if (c == arr[i]) return true;

}

return false;

}

//Удаление знаков препинания, не разделенных от слов пробелами

static string PunctuationDel(string s)

{

char[] beginning = { '(', '<', '\'', '\"', '«', '•', '◆', '⇨', '́' };

char[] end = { ';', '.', ',', '\'', '\"', ':', '!', '?', '>', ')', '%', '»', '•', '◆', '⇨', '́' };

while (SymbExistance(beginning, s[0]))

{

s = s.Substring(1, s.Length - 1);

}

while (SymbExistance(end, s[s.Length - 1]))

{

s = s.Substring(0, s.Length - 1);

}

return s;

}

//Формирование списка уникальных основ в тексте с указанием их частот

public static List<Word> Frequency(List<string> text)

{

int index = 0;

List<Word> Freq = new List<Word>();

for (int i = 0; i < text.Count; i++)

{

if (FreqCheck(Freq, text[i], ref index))

{

Freq[index].count++;

}

else

{

Freq.Add(new Word(text[i]));

}

text.Remove(text[i]);

i--;

}

return Freq;

}

//Проверка на повторение слова

static bool FreqCheck(List<Word> txt, string w, ref int ind)

{

for (int i = 0; i < txt.Count; i++)

{

if (txt[i].word == w)

{

ind = i;

return true;

}

}

return false;

}

static void LLWchange()

{

string[] wordlist;

List<List<Word>> temp = new List<List<Word>>();

AllWds = AllWords();

wordlist = new string[AllWds.Count];

for (int i = 0; i < AllWds.Count; i++)

{

wordlist[i] = AllWds[i];

}

for (int i = 0; i < Tag.Count; i++)

{

temp.Add(new List<Word>());

for (int j = 0; j < wordlist.Length; j++)

{

bool wordexists = false;

for (int k = 0; k < Tag[i].Count; k++)

{

if (wordlist[j] == Tag[i][k].word)

{

wordexists = true;

temp[i].Add(new Word(Tag[i][k].word, Tag[i][k].count));

break;

}

}

if (!wordexists)

{

temp[i].Add(new Word(wordlist[j], 0));

}

}

}

Tag = temp;

}

//Формирование списка уникальных основ слов, которые повторяются в РАЗНЫХ текстах

static List<string> AllWords()

{

List<string> ss = new List<string>();

List<Word> temp = new List<Word>();

for (int i = 0; i < Tag.Count; i++)

{

for (int j = 0; j < Tag[i].Count; j++)

{

if (!CheckItemInList(ref temp, Tag[i][j].word))

{

temp.Add(new Word(Tag[i][j].word));

}

}

}

for (int i = 0; i < temp.Count; i++)

{

if (!temp[i].IsSingle) ss.Add(temp[i].word);

}

return ss;

}

//Проверка на наличие элемента в списке

static bool CheckItemInList(ref List<Word> ss, string elem)

{

for (int i = 0; i < ss.Count; i++)

{

if (ss[i].word == elem)

{

ss[i].IsSingle = false;

return true;

}

}

return false;

}

//Формирование частотной матрицы

public static void FormMatrix()

{

LLWchange();

Matrix = new double[Tag[0].Count, Tag.Count];

TextTitles = new Tags[Matrix.GetLength(1)];

Words = new Tags[Matrix.GetLength(0)];

for (int i = 0; i < Tag[0].Count; i++)

{

for (int j = 0; j < Tag.Count; j++)

{

Matrix[i, j] = Tag[j][i].count;

TextTitles[j] = new TextTitle(Convert.ToString(j + 1));

Words[i] = Tag[j][i];

}

}

}

//Проверка слова на принадлежность к языку

public static int LanguageCheck(string word, out string wrd)

{

int rez = 0;

string temp = "";

for (int i = 0; i < word.Length; i++)

{

if (word[i] == 'Ё' || word[i] == 'ё') temp += "е";

else if (((int)(word[i]) >= 1040 && (int)(word[i]) <= 1071) || ((int)(word[i]) >= 65 && (int)(word[i]) <= 90))

{ temp += Convert.ToString(Convert.ToChar(Convert.ToInt32(word[i]) + 32)); }

else temp += Convert.ToString(word[i]);

if ((int)temp[temp.Length - 1] >= 1072 && (int)temp[temp.Length - 1] <= 1103)

{

if (rez == 0 || rez == 1) rez = 1;

else if (rez == 2) { rez = 0; temp = word; break; }

}

else if ((int)temp[temp.Length - 1] >= 97 && (int)temp[temp.Length - 1] <= 122)

{

if (rez == 0 || rez == 2) rez = 2;

else if (rez == 1) { rez = 0; break; }

}

}

wrd = temp;

return rez;

//0 - кириллица и латынь (очепятка)

//1 - кириллица (русский язык)

//2 - латиница (английский язык)

}

public static double[,] GetMatrix

{

get { return Matrix; }

}

public static Tags[] GetWords

{

get { return Words; }

}

public static Tags[] GetTextTitles

{

get { return TextTitles; }

}

}

class TextTitle:Tags

{

public string Title;

public TextTitle(string Title)

{

this.Title = Title;

}

public override string GetTag

{

get

{

return Title;

}

set

{

Title = value;

}

}

}

public class Vertex

{

public float t, p; //временный и постоянный приоритетные коэффициенты

public int cocheck;

public double x, y, z;

public bool isin, wasMin;

public Vertex prev; // предыдущая вершина в минимальном остовном дереве/графе минимального пути

public Tags Data;

protected int rad;

protected static int ax = 0;

protected static int ay = 0;

public Vertex(Tags Data, double x, double y, double z)

{

this.x = x;

this.y = y;

this.z = z;

this.Data = Data;

t = float.MaxValue;

p = float.MaxValue;

prev = null;

isin = false;

wasMin = false;

rad = 7;

}

public void Draw(Graphics g)

{

Brush br;

if (Data is Word) br = new SolidBrush(Color.Red);

else br = new SolidBrush(Color.Blue);

g.FillEllipse(br, (int)(x \* Form2.GetScale) - rad + Form1.GetForm2Width / 2 + ax, (int)(y \* Form2.GetScale) - rad + Form1.GetForm2Height / 2 + ay, rad \* 2, rad \* 2);

g.DrawEllipse(new Pen(new SolidBrush(Color.Black)), (int)(x \* Form2.GetScale) - rad + Form1.GetForm2Width / 2 + ax, (int)(y \* Form2.GetScale) - rad + Form1.GetForm2Height / 2 + ay, rad \* 2, rad \* 2);

}

public bool Clicked(float x0, float y0)

{

if (Math.Pow(x0 - ((x \* Form2.GetScale) + Form1.GetForm2Width / 2 + ax), 2) + Math.Pow(y0 - ((y \* Form2.GetScale) + Form1.GetForm2Height / 2 + ay), 2) <= Math.Pow(rad, 2))

{

return true;

}

else return false;

}

public int GetCoordX

{

get { return (int)(x \* Form2.GetScale) - rad + Form1.GetForm2Width / 2 + ax; }

}

public int GetCoordY

{

get { return (int)(y \* Form2.GetScale) - rad + Form1.GetForm2Height / 2 + ay; }

}

public int GetRad

{

get { return rad; }

}

public static int AX

{

get { return ax; }

set { ax = value; }

}

public static int AY

{

get { return ay; }

set { ay = value; }

}

}

public class Word:Tags

{

public string word;

public int count;

public bool IsSingle;

public Word(string word)

{

this.word = word;

count = 1;

IsSingle = true;

}

public Word(string word, int c)

{

this.word = word;

count = c;

}

public Word(string s, bool IsSingle)

{

this.IsSingle = IsSingle;

this.word = s;

}

public override string GetTag

{

get

{

return word;

}

set

{

word = value;

}

}

}

public partial class Form1 : Form

{

Form2 F;

string[] t;

public static bool usertxtentered;

List<string> input;

static int formwidth, formheight;

public static int ftn;

public Form1()

{

TestData.SetData();

InitializeComponent();

usertxtentered = false;

input = new List<string>();

F = new Form2();

formwidth = F.Size.Width;

formheight = F.Size.Height;

t = new string[]{ "Андре Александрович Мирон (фамил при — Мена́кер[1]; 7 март 1941, Москв — 16 август 1987, Рига) — советск актёр театр и кино, артист эстрады. Народн артист РСФСР (1980).",

"Евген Витальевич Мирон (род. 29 ноябр 1966, Саратов, СССР) — советск и российск актёр театр и кино, народн артист Росс (2004)[1], лауреат двух Государствен прем Российск Федерации.",

"Серг Михайлович Мирон (14 феврал 1953, Пушкин, Ленинград) — российск политическ и государствен деятель, депутат Государствен дум VI созыва, руководител фракц парт «Справедлив Россия» в Государствен думе, председател совет Палат депутат парт «Справедлив Россия» — член бюр президиум Центральн совет парт (2011—2013). Ран — депутат Государствен дум V созыв (2011), Председател Совет Федерац (2001—2011), депутат Законодательн собран Санкт-Петербург (1994—2001). Председател парт «Справедлив Россия» в 2006—2011 и с 27 октябр 2013 года, ран — председател Российск парт Жизни. Выставля сво кандидатур на выбор президент РФ в 2004 и 2012 год и об раз занима последн место. Председател Наблюдательн совет «Союз десантник России».",

"Руководител фракц «Справедлив Россия» в Госдум Серг Мирон предлож запрет реклам кредит и займ посредств наружн рекламы, SMS-рассылок и проч носителей. Политик подготов соответств поправк в КоАП, пишет газет «Известия». Согласн документу, за распространен реклам кредит и займ предусмотр штраф. Для физическ лиц ег величин состав до 5 тысяч рублей; для индивидуальн предпринимател — до 50 тысяч рублей, для юрлиц — до 1 миллион рублей. Банк смогут рекламирова сво услуг тольк в собствен офисах. Как говор в пояснительн записк к законопроекту, в кризис «широк реклам потребительск кредит посредств наружн рекламы, телевизион и радиовещания, SMS-рассылок, листовок, объявлен и т.д. приобрел характер массов социальн провокации». При эт «невозможн погашен населен кредит стал существен фактор не тольк экономическ проблем, но и повышен социальн напряжен в целом», отмеча автор. По мнен Миронова, запрет на реклам кредит «поможет уменьш количеств иск о невозвращен кредитах, сниз ставк по потребительск кредитам, увелич социальн ответствен гражда и укреп их финансов независимость». Он добавил, что решен гражда об обращен за кредит «должн быт хорош обдуманным, принят с должн мер ответственности, а не явля прям следств агрессивн рекламн кампаний». В комитет по финансов рынку, по слов член эт комитет коммунист Борис Кашина, поддержа законопроект пок не готовы.",

"Актер, Режиссер рост 1.82 м 8 марта, 1941 • рыб рыб Москва, СССР (Россия) дат смерт 16 авгуcта, 1987 • 46 лет Рига, СССР (Латвия) жанр комедия, драма, мелодрам супруг Екатерин Градов (развод) ... один ребенок Ларис Голубкин всег фильм 70, 1962 — 1993",

"Актер, Продюсер, Актер: Дубляж, Режиссер, Сценарист рост 1.73 м 29 ноября, 1966 • стрелец стрелец • 48 лет Татищево, Саратовск область, СССР (Россия) жанр драма, комедия, воен всег фильм 72, 1987 — 2017",

};

TextOperations.InitParams(t);

}

private void button1\_Click(object sender, EventArgs e)

{

if (richTextBox1.Text != "")

{

usertxtentered = true;

input.Add(richTextBox1.Text);

TextOperations.InitParams(input);

listBox1.Items.Add(TextOperations.ts[TextOperations.ts.Count - 1]);

richTextBox1.Text = "";

}

}

private void button2\_Click(object sender, EventArgs e)

{

F = new Form2();

if (usertxtentered && input.Count >1)

{

string[] temp = new string[input.Count];

for (int i = 0; i < temp.Length; i++)

{

temp[i] = input[i];

}

TestData.Texts.Add(temp);

}

TextOperations.TagNullifier();

for (int i = 0; i < TextOperations.ts.Count; i++)

{

TextOperations.Tag.Add(TextOperations.Frequency(TextOperations.vClusterize(TextOperations.ts[i].ToString())));

}

TextOperations.FormMatrix();

F.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

}

public static int GetForm2Width

{

get { return formwidth; }

set { formwidth = value; }

}

public static int GetForm2Height

{

get { return formheight; }

set { formheight = value; }

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

}

void UpdateForm2Params(object sender, EventArgs e)

{

formheight = F.Size.Height;

formwidth = F.Size.Width;

}

private void button3\_Click\_1(object sender, EventArgs e)

{

input = new List<string>();

listBox1.Items.Clear();

usertxtentered = false;

TestData.Texts.Remove(TestData.Texts[TestData.Texts.Count - 1]);

}

}

public partial class Form2 : Form

{

Clusters C;

int mousex, mousey;

static int scale = 400;

bool moving;

PointF movestart;

bool EnableClusterPainting;

Form3 ParScroll;

string kstring;

string lstring;

public static int limit;

public Form2()

{

InitializeComponent();

this.MouseWheel += new MouseEventHandler(Form2\_MouseWheel);

EnableClusterPainting = false;

mousex = 0;

mousey = 0;

label2.BackColor = Color.Transparent;

moving = false;

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

movestart = new PointF();

ParScroll = new Form3();

kstring = textBox1.Text;

lstring = textBox2.Text;

limit = Convert.ToInt32(lstring);

if (Form1.usertxtentered)

{

comboBox2.Items.Add("Пользовательские данные ");

}

}

private void CBaction()

{

double k = Convert.ToDouble(textBox1.Text);

limit = Convert.ToInt32(textBox2.Text);

if (comboBox1.SelectedIndex == 0)

{

C.ClusterSelection(k, 2);

}

else

{

if (comboBox1.SelectedIndex == 1)

{

C.ClusterSelection(k, 3);

}

else

{

if (comboBox1.SelectedIndex == 2)

{

C.ClusterSelection(k, 1);

}

else

{

if (comboBox1.SelectedIndex == 3)

{

C.ClusterSelection(k, 4);

}

else

{

C.ClusterSelection(k, 5);

}

}

}

}

}

private void UpdateParameter(object sender, ParameterEventArgs e)

{

textBox1.Text = Convert.ToString(e.GetPar);

textBox3.Clear();

TextOperations.InitParams(TestData.Texts[comboBox2.SelectedIndex]);

TextOperations.TagNullifier();

for (int i = 0; i < TextOperations.ts.Count; i++)

{

TextOperations.Tag.Add(TextOperations.Frequency(TextOperations.vClusterize(TextOperations.ts[i].ToString())));

}

TextOperations.FormMatrix();

if (EnableClusterPainting)

{

C = new Clusters(TextOperations.GetMatrix, TextOperations.GetTextTitles, TextOperations.GetWords);

CBaction();

textBox3.Clear();

for (int i = 0; i < C.GetClusters.Count; i++)

{

for (int j = 0; j < C.GetClusters[i].Data.Count; j++)

{

textBox3.Text += C.GetClusters[i].Data[j].Data.GetTag += " ";

}

textBox3.Text += " | ";

}

Invalidate();

}

}

private void Visualization\_Load(object sender, EventArgs e)

{

DoubleBuffered = true;

}

private void button1\_Click(object sender, EventArgs e)

{

textBox3.Clear();

C = new Clusters(TextOperations.GetMatrix, TextOperations.GetTextTitles, TextOperations.GetWords);

CBaction();

EnableClusterPainting = true;

textBox3.Clear();

for (int i = 0; i < C.GetClusters.Count; i++)

{

for (int j = 0; j < C.GetClusters[i].Data.Count; j++)

{

textBox3.Text += C.GetClusters[i].Data[j].Data.GetTag += " ";

}

textBox3.Text += " | ";

}

Invalidate();

}

private void button1\_Paint(object sender, PaintEventArgs e)

{

}

private void Form2\_Paint(object sender, PaintEventArgs e)

{

if (EnableClusterPainting)

{

e.Graphics.Clear(this.BackColor);

for (int i = 0; i < C.GetGraph.E.Count; i++)

{

C.GetGraph.E[i].Draw(e.Graphics);

}

for (int i = 0; i < C.GetGraph.V.Count; i++)

{

C.GetGraph.V[i].Draw(e.Graphics);

}

for (int i = 0; i < C.GetClusters.Count; i++)

{

C.GetClusters[i].ConvexHull(e.Graphics);

}

}

}

private void Form2\_SizeChanged(object sender, EventArgs e)

{

EventArgs par = new EventArgs();

}

private void Form2\_MouseHover(object sender, EventArgs e)

{

}

private void Form2\_MouseMove(object sender, MouseEventArgs e)

{

if (moving)

{

Vertex.AX += e.X - (int)movestart.X;

Vertex.AY += e.Y - (int)movestart.Y;

movestart.X = e.X;

movestart.Y = e.Y;

Invalidate();

}

else

{

if (EnableClusterPainting)

{

bool vselected = false;

label2.Text = "";

for (int i = 0; i < C.GetGraph.V.Count; i++)

{

if (C.GetGraph.V[i].Clicked(e.X, e.Y))

{

mousex = e.X;

mousey = e.Y;

label2.Visible = true;

vselected = true;

label2.Text += " ";

label2.Text += C.GetGraph.V[i].Data.GetTag;

label2.Left = C.GetGraph.V[i].GetCoordX + 20;

label2.Top = C.GetGraph.V[i].GetCoordY + 20;

}

}

if (!vselected) label2.Visible = false;

}

}

}

private void Form2\_FormClosed(object sender, FormClosedEventArgs e)

{

EnableClusterPainting = false;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

public static int GetScale

{

get { return scale; }

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void Form2\_MouseDown(object sender, MouseEventArgs e)

{

if (EnableClusterPainting)

{

moving = true;

movestart.X = e.X;

movestart.Y = e.Y;

}

}

private void Form2\_MouseUp(object sender, MouseEventArgs e)

{

moving = false;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (EnableClusterPainting)

{

C = new Clusters(TextOperations.GetMatrix, TextOperations.GetTextTitles, TextOperations.GetWords);

CBaction();

textBox3.Clear();

for (int i = 0; i < C.GetClusters.Count; i++)

{

for (int j = 0; j < C.GetClusters[i].Data.Count; j++)

{

textBox3.Text += C.GetClusters[i].Data[j].Data.GetTag += " ";

}

textBox3.Text += " | ";

}

Invalidate();

}

}

private void Form2\_MouseWheel(object sender, MouseEventArgs e)

{

if (e.Delta >= 120)

{

scale += 100;

}

else

{

if (e.Delta <= -120&&scale>=100)

{

scale -= 100;

}

}

Invalidate();

}

private void comboBox1\_MouseWheel(object sender, MouseEventArgs e)

{

}

private void Form2\_MouseEnter(object sender, EventArgs e)

{

}

private void comboBox1\_MouseLeave(object sender, EventArgs e)

{

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

textBox3.Clear();

TextOperations.InitParams(TestData.Texts[comboBox2.SelectedIndex]);

TextOperations.TagNullifier();

for (int i = 0; i < TextOperations.ts.Count; i++)

{

TextOperations.Tag.Add(TextOperations.Frequency(TextOperations.vClusterize(TextOperations.ts[i].ToString())));

}

TextOperations.FormMatrix();

if (EnableClusterPainting)

{

C = new Clusters(TextOperations.GetMatrix, TextOperations.GetTextTitles, TextOperations.GetWords);

CBaction();

textBox3.Clear();

for (int i = 0; i < C.GetClusters.Count; i++)

{

for (int j = 0; j < C.GetClusters[i].Data.Count; j++)

{

textBox3.Text += C.GetClusters[i].Data[j].Data.GetTag += " ";

}

textBox3.Text += " | ";

}

Invalidate();

}

}

private void button2\_Click\_1(object sender, EventArgs e)

{

if (ParScroll.Shw)

{

ParScroll.PCevent += new ParameterChangedEvents(this.UpdateParameter);

ParScroll.Activate();

ParScroll.WindowState = FormWindowState.Normal;

}

else

{

ParScroll = new Form3();

ParScroll.K = Convert.ToDouble(textBox1.Text);

ParScroll.PCevent += new ParameterChangedEvents(this.UpdateParameter);

ParScroll.Show();

ParScroll.Shw = true;

}

}

private void Form2\_Activated(object sender, EventArgs e)

{

ParScroll.TBactive = false;

}

private void comboBox1\_DropDownClosed(object sender, EventArgs e)

{

label1.Focus();

}

private void comboBox2\_DropDownClosed(object sender, EventArgs e)

{

label1.Focus();

}

private void textBox2\_TextChanged\_1(object sender, EventArgs e)

{

bool error = false;

int k;

try

{

if (textBox2.Text != "")

{

button1.Enabled = true;

button2.Enabled = true;

k = Convert.ToInt32(textBox2.Text);

}

else

{

button1.Enabled = false;

button2.Enabled = false;

}

}

catch (System.Exception)

{

error = true;

MessageBox.Show("Невертый формат текста");

textBox2.Text = lstring;

}

if (!error)

{

lstring = textBox2.Text;

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

bool error = false;

double k;

try

{

if (textBox1.Text != "")

{

button1.Enabled = true;

button2.Enabled = true;

k = Convert.ToDouble(textBox1.Text);

}

else

{

button1.Enabled = false;

button2.Enabled = false;

}

}

catch (System.Exception)

{

error = true;

MessageBox.Show("Невертый формат текста");

textBox1.Text = kstring;

}

if (!error)

{

kstring = textBox1.Text;

}

}

private void Form2\_MouseDoubleClick(object sender, MouseEventArgs e)

{

if (EnableClusterPainting)

{

for (int i = 0; i < C.GetGraph.V.Count; i++)

{

if (C.GetGraph.V[i].Clicked(e.X, e.Y) && C.GetGraph.V[i].Data is TextTitle)

{

MessageBox.Show(TestData.Texts[comboBox2.SelectedIndex][Convert.ToInt32(C.GetGraph.V[i].Data.GetTag) - 1]);

}

}

}

}

private void textBox2\_TextChanged\_2(object sender, EventArgs e)

{

}

}

public partial class Form3 : Form

{

bool isopened;

bool tbactive;

public event ParameterChangedEvents PCevent;

public Form3()

{

InitializeComponent();

}

public bool Shw

{

get

{

return isopened;

}

set

{

isopened = value;

}

}

public bool TBactive

{

get { return tbactive; }

set { tbactive = value; }

}

public double K

{

set { trackBar1.Value = Convert.ToInt32(value \* 10); }

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

ParameterEventArgs p = new ParameterEventArgs(Convert.ToDouble(trackBar1.Value) / 10);

PCevent(this, p);

}

private void Form3\_FormClosing(object sender, FormClosingEventArgs e)

{

isopened = false;

}

private void Form3\_Activated(object sender, EventArgs e)

{

tbactive = true;

}

}

public class StWdsEng

{

public string[] Eng = {

"1","2","3","4","5","6","7","8","9","0",

"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t",

"u","v","w","x","y","z",

"about","above","according","across","actually","ad","adj","ae","af","after","afterwards",

"ag","again","against","ai","al","all","almost","alone","along","already","also","although",

"always","am","among","amongst","an","and","another","any","anyhow","anyone","anything","anywhere","ao","aq",

"ba","bb","bd","be","became","because","become","becomes","becoming","been","before","beforehand","begin","beginning",

"behind","being","below","beside","besides","between","beyond","bf","bg","bh","bi","billion","bj","bm","bn","bo","both",

"br","bs","bt","but","buy","bv","bw","by","bz","ca","can","can't","cannot","caption","cc","cd","cf","cg","ch","ci",

"ck","cl","click","cm","cn","co","co.","com","copy","could","couldn","couldn't","cr","cs","cu","cv","cx","cy","cz",

"de","did","didn","didn't","dj","dk","dm","do","does","doesn","doesn't","don","don't","down","during","dz",

"each","ec","edu","ee","eg","eh","eight","eighty","either","else","elsewhere","end","ending","enough","er",

"es","et","etc","even","ever","every","everyone","everything","everywhere","except",

"few","fi","fifty","find","first","five","fj","fk","fm","fo","for","former","formerly","forty","found","four","fr","free","from","further","fx",

"ga","gb","gd","ge","get","gf","gg","gh","gi","gl","gm","gmt","gn","go","gov","gp","gq","gr","gs","gt","gu","gw","gy",

"had","has","hasn","hasn't","have","haven","haven't","he","he'd","he'll","he's",

"help","hence","her","here","here's","hereafter","hereby","herein","hereupon","hers","herself","him","himself",

"his","hk","hm","hn","home","homepage","how","however","hr","ht","htm","html","http","hu","hundred",

"i'd","i'll","i'm","i've","i.e.","id","ie","if","ii","il","im","in","inc","inc."

,"indeed","information","instead","int","into","io","iq","ir","is","isn","isn't"

,"it","it's","its","itself",

"je","jm","jo","join","jp",

"ke","kg","kh","ki","km","kn","koo","kp","kr","kw","ky","kz",

"la","last","later","latter","lb","lc","least","less","let","let's","li","like",

"likely","lk","ll","lr","ls","lt","ltd","lu","lv","ly",

"ma","made","make","makes","many","maybe","mc","md","me","meantime","meanwhile",

"mg","mh","microsoft","might","mil","million","miss","mk","ml","mm","mn","mo",

"more","moreover","most","mostly","mp","mq","mr","mrs","ms","msie","mt","mu","much","must","mv","mw","mx","my","myself","mz",

"na","namely","nc","ne","neither","net","netscape","never","nevertheless","new",

"next","nf","ng","ni","nine","ninety","nl","no","nobody","none","nonetheless","noone",

"nor","not","nothing","now","nowhere","np","nr","nu","null","nz",

"of","off","often","om","on","once","one","one's","only","onto","or","org","other",

"others","otherwise","our","ours","ourselves","out","over","overall","own",

"pa","page","pe","per","perhaps","pf","pg","ph","pk","pl","pm","pn","pr","pt","pw","py",

"qa",

"rather","re","recent","recently","reserved","ring","ro","ru","rw",

"sa","same","sb","sc","sd","se","seem","seemed","seeming","seems","seven","seventy"

,"several","sg","sh","she","she'd","she'll","she's","should","shouldn","shouldn't",

"si","since","site","six","sixty","sj","sk","sl","sm","sn","so","some","somehow",

"someone","something","sometime","sometimes",

"taking","tc","td","ten","text","tf","tg","test","th","than","that","that'll","that's",

"the","their","them","themselves","then","thence","there","there'll","there's",

"thereafter","thereby","therefore","therein","thereupon","these","they","they'd",

"they'll","they're","they've","thirty","t", "to",

"ua","ug","uk","um","under","unless","unlike","unlikely","until","up","upon","us","use","used","using","uy","uz",

"va","vc","ve","very","vg","vi","via","vn","vu",

"was","wasn","wasn't","we","we'd","we'll","we're","we've","web","webpage","website",

"welcome","well","were","weren","weren't","wf","what","what'll","what's","whatever",

"when","whence","whenever","where","whereafter","whereas","whereby","wherein","whereupon","wherever","whether","which",

"ye","yes","yet","you","you'd","you'll","you're","you've","your","yours","yourself","yourselves","yt","yu",

"za","zm","zr"

};

public bool Check(string s)

{

foreach (string elem in Eng)

{

if (s == elem) return true;

}

return false;

}

}

public class StWdsRus

{

public string[] Rus = {

"1","2","3","4","5","6","7","8","9","0",

"а",

"б", "без","буд", "больш", "был","бы",

"в", "вам", "вас","во","вс","всех", "всегда", "ведь", "вот", "впрочем", "вдруг", "вмест",

"где", "говор",

"даже","до", "для", "да", "дел",

"ее","есть","ему", "его", "ее", "ей", "если", "еще", "ег","ест",

"ж", "же",

"здес", "зач", "з", "занима","за",

"из","им", "и", "из-за", "или", "им", "иногд","ил",

"к","ко","кто", "как", "когд", "котор", "куд",

"либо", "ли",

"может", "между", "мен", "мне", "мног", "мож", "мог", "можн", "м","мы",

"но","наш","него","нет","них","ну", "надо", "наконец", "нас", "н", "нибуд", "ним", "ни", "не", "на", "называ", "назван", "назва", "ними","например","нам","нача",

"о","от", "он", "обычн", "основа", "об",

"по","при", "перед", "пред", "под", "про", "побыва","п",

"род",

"со", "с", "сам", "св", "сво", "своих", "себ", "сказа", "скаж", "стат", "стал", "станов",

"тем", "т", "так", "теб", "тепер", "там", "тот", "тех", "тут", "то","том",

"у","уж",

"ч","чтоб", "что", "част","чем",

"эт", "этот",

"я", "явля"};

public bool Check(string s)

{

foreach (string elem in Rus)

{

if (s == elem) return true;

}

return false;

}

}