МРНТИ 20.23.17; 20.23.21
УДК 004.421, 004.912

https://doi.org/10.51889/2020-2.1728-7901.45

Д.Р. Рахимова[1], А.Р. Сатыбалдиев[1]

[1]Казахский Национальный Университет им. Аль-Фараби, г. Алматы, Казахстан

АЛГОРИТМ СБОРА ТЕКСТОВЫХ ДАННЫХ НА КАЗАХСКОМ ЯЗЫКЕ

Аннотация

Работа посвящена созданию системы автоматического сбора и обработки открытых данных на казахском языке с ресурсов сети интернет, и несет в себе практическую значимость в задачах сбора и анализа текста. Во введении обосновывается актуальность выбранной темы, обзор существующих подходов, формулируются задачи исследования. Рассматривается такая задача, как сбор и первичная обработка текстовых данных с последующим анализом. Сбор данных является первоочередной задачей, так как открытые данные с ресурсов сети интернет не структурированы и нуждаются в обработке. Авторы предоставляют систему обработки веб страниц казахскоязычных порталов, а также приводят практическое применение данного подхода на реальных данных открытых ресурсов с помощью созданной системы. Представлен подход индексирования документов с помощью признаков**.** Система поможет структурировать открытые данные с ресурсов сети интернет, а также провести анализ собранных данных. Представлены практические результаты.

Ключевые слова: алгоритм, сбор данных, текстовый анализ, казахский язык**.**

Аңдатпа

Д. Р. Рахимова[1], А. Р. Сатыбалдиев[1]

[1]Әл-Фараби атындағы Қазақ Ұлттық Университеті, Алматы қ., Қазақстан.

ҚАЗАҚ ТІЛІНДЕГІ МӘТІН ДЕРЕКТЕРДІ ЖИНАУ АЛГОРИТМІ

Бұл ғылыми жұмыс заманауи технологияларды пайдаланып, интернет-ресурстардан қазақ тіліндегі ашық деректерді автоматты түрде жинау және өңдеу жүйесін құруға арналған. Осы жұмыста көрсетілген нәтижелер мәтіндік деректерді жинау және талдау тапсырмаларында практикалық маңызы бар. Кіріспе бөлімінде таңдалған тақырыптың өзектілігін, қолданыстағы тәсілдерге шолу жасайды, зерттеу мақсаттарын тұжырымдайды. Біз мәтіндік деректерді кейінгі талдау арқылы жинау және бастапқы өңдеу сияқты проблеманы қарастырамыз. Деректерді жинау алгоритмі басымдық болып табылады, өйткені интернет-ресурстардың ашық деректері құрылымдалмаған түрде болып келеді және оларды өңдеу қажет. Авторлар қазақ тілдік порталдардың веб-парақтарын өңдеудің жүйесін ұсынады, сонымен қатар құрылған жүйені қолдана отырып, құжаттарды индекстеу тәсілдері арқылы деректердерді сұрыптайды. Жасалған тәсілдің практикалық қолданылуы келтірген.

Түйін сөздер: алгоритм, деректі жинау, мәтінді талдау, қазақ тілі.

Abstract

ALGORITHM COLLECTION OF TEXT DATA IN THE KAZAKH LANGUAGE

Rakhimova D.R.[1], Satybaldiev A.R.[1]

[1]Al-Farabi Kazakh National University, Almaty, Kazakhstan

This work is devoted to the creation of a system for the automatic collection and processing of open data in Kazakh from Internet resources, and bears practical significance in the tasks of collecting and analyzing text. The introduction substantiates the relevance of the chosen topic, a review of existing approaches, formulates the objectives of the study. We consider such a problem as the collection and primary processing of text data with subsequent analysis. Data collection is a priority, since open data from Internet resources is not structured and needs to be processed. The authors provide a system for processing web pages of Kazakh-language portals, and also gives practical application of this approach to real data of open resources using the created system. The approach of indexing documents using features is presented. The system will help structure open data from Internet resources, as well as analyze collected data. Practical results are presented.

Keywords: algorithm, data collection, text analysis, Kazakh language.

Введение

Важность, значимость и необходимость анализа и обработки текстовых и других слабоструктурированных информационных данных постоянно возрастают. В связи с широким распространением систем электронного документооборота, социальных сетей, блогов, сетевых информационных порталов, персональных сайтов это становится особенно важным и как

техническая задача, и как значимая часть взаимодействия людей в современном информационном мире.

Одной из основных форм представления информации является текстовая форма, наряду с графической, звуковой, а также видео информацией. Если первоначально первостепенными проблемами считались задачи, связанные с обеспечением сбора, хранения, поиска и предоставления данных, то в последнее время, при упрощении доступа к разнообразным коллекциям текстовых документов, появляются новые задачи анализа и обработки текстовых данных. К традиционным проблемам добавляются новые, связанные, например, с большими объемами текстовых данных в различных социальных сетях и других информационных, поисковых и аналитических приложениях Интернета [1].

Обзор существуещих подходов

Для исследования и решения различных интеллектуальных задач связанных в вычислительной лингвистике основной задачей является сбор и анализ электронных текстовых данных на различных языках. Для быстрого сбора данных существуют подход, основанный на принципе поисковых пауков. В результате нашего исследования были отобраны 3 наиболее актуальных на сегодняшний день реализации поисковых пауков: Scrapy, Heritrix и Apache Nutch. Был проведен сравнительный анализ и разработан собственный поисковый паук.

Scrapy – это фреймворк для языка Python, который хорошо подходит для задач быстрого одноразового поиска. Scrapy легко настраивается и быстро работает с маленьким количеством источников данных. Фреймворк не имеет встроенной функциональности для работы в распределенном режиме. Scrapy не поддерживает ключевые требования, предъявляемые к разрабатываемой системе: возможности работать в распределённой среде и масштабируемости [2].

Heritrix – это полноценный поисковый робот, написанный на языке Java. Он поддерживает работу в распределенной среде, но не позволяет гибко масштабировать систему. Добавление нового узла требует остановки и конфигурации всей системы. Также система, построенная на основе Heritrix, ненадежная и не имеет механизмов отказоустойчивости, сбой одного из узлов во время работы приводит к потере данных [3].

Apache Nutch – это поисковый паук с открытым исходным кодом, написанный на Java. Основным преимуществом системы Apache Nutch является ее интеграция с экосистемой Apache Hadoop и ее распределенной файловой системой HDFS. Благодаря этому Nutch получает все преимущества Hadoop: отказоустойчивость, надежность и распределенность. Также Apache Nutch легко интегрируется с системами индексации данных, такими как Apache Solr и Apache Lucene [4].

Алгоритм сбора текстовых данных

При создании алгоритма автоматического пополнения текстов были выделены основные модули (функциональные блоки):

1) *Подготовка (Initialization):* отвечает за аутентификацию, получение cookies, распознаванию captcha, кэшированию записей DNS и т.д.;

2) *Загрузка (Download):* выполняет загрузку определённой страницы;

3) *Проверка (Check):* проверка корректности загрузки, например, на размер страницы, timeout, ошибку 404 и т.п.;

4) *Запрос (Requests)* если у базовой страницы только одна страница с нужными данными, то идём в блок Requests. А если у базовой страницы есть подстраницы, то попадаем в блок Under the pages. Там собираем все нужные ссылки, и только потом попадаем в блок Requests.

5) *Разбор (Parse):* разбор страницы согласно правилам;

6) *Извлечение (Extraction):* получение данных из разобранной страницы, возможно, с использованием дополнительных справочников и данных;

7) *Обновление (Update):* обновление данных в БД;

8) *Контроль (Monitoring):* обработка данных других модулей, ведение журналов, предоставление информации о текущем статусе.

Хочется подчеркнуть важность первого и третьего блоков. В блоке подготовки (Initialization) выполняются все операции, которые необходимы для начала работы с конкретным сервером, а также обработка «нестандартных» страниц. Возможно, что перед получением страниц с данными необходимо выполнить port knocking, загрузить определённую страницу (или картинку) для получения cookie, доказать, что страницы посещает человек (captcha, контрольный вопрос). Здесь же определяются и создаются потоки и очередь для конкретного сервера. Фактически, данный модуль может принимать решения о приостановке загрузки, а также её возобновления после устранения

причины ошибки. В рамках системы данный блок должен быть один для каждого сайта, поскольку, например, повторная аутентификация может привести к сбросу сессии.

Модуль проверки необходим для оперативной оценки корректности загрузки конкретной страницы. Хотя по времени он работает непосредственно сразу после блока загрузки (Download), но часто имеет индивидуальную реализацию для каждого сайта. Корректная страница не означает, что там находится именно то, что нужно, поскольку сайт может сообщать об ошибке авторизации, необходимости пройти аутентификацию, повторить попытку через некоторое время, невозможности найти подходящую страницу и т.п. В качестве критериев распознавания можно использовать сокращённый модуль разбора (Parse), но дополнительно необходимо обрабатывать сообщения об ошибках. Также друг от друга отделяется блок разбора и блок экстракции. Причина этого в том, что целью разбора является анализ страницы и выделение фрагментов с данными, а экстракция уже привязывает данные к конкретным объектам. Разбор работает на уровне HTML и ограничен конкретной страницей, а блок экстракции работает уже с данными и может воспользоваться информацией ранее загруженных страниц. У блока обновления задача осложняется тем, что данные надо объединять, причём они могут содержать различное число полей и быть корректными лишь на определённые моменты времени. Целью модуля контроля является не только получение информации о текущем статусе и процессе выполнения операций, но и выявление сбоев и возможность перезапуска с любого этапа.

Для сбора данных в основном были акцентированы несколько информационных порталов, такие как akorda.kz, nur.kz, wiki.py и др. Для разработки алгоритма автоматического пополнения корпуса использовали язык Python с расширением специальных библиотек:

1) *Requests* - это библиотека Python HTTP, выпущенная под лицензией Apache2. Цель проекта - сделать запросы HTTP проще и удобнее для человека. Текущая версия 2.22.0.

2) *Beautiful Soup 4* - это пакет Python для анализа документов HTML и XML. Он создает дерево синтаксического анализа для проанализированных страниц, которое можно использовать для извлечения данных из HTML, что полезно для просмотра веб-страниц. Он доступен для Python 2.7 и Python 3 (для этого реализации данных задач использовано 4 версия).

3) *lxml* – это библиотека, которая позволяет легко обрабатывать XML и HTML файлы, а также может использоваться для парсинга веб-страниц. Существует множество готовых парсеров XML/HTML, но для получения лучших результатов или при определенных задачах разработчики вынуждены писать свои собственные парсеры. Это как раз та ситуация, когда возникает необходимость в lxml библиотеке. Ключевые преимущества этой библиотеки заключаются в том, что она проста в использовании, чрезвычайно быстра при анализе больших документов, очень хорошо документирована и обеспечивает простое преобразование исходных данных в типы данных Python, что упрощает манипулирование файлами.

4) *PyMySQL* - это клиентская библиотека Python MySQL, основанная на PEP 249. Большинство общедоступных API совместимы с mysqlclient и MySQLdb. PyMySQL работает с MySQL 5.5+ и MariaDB 5.5+. MySQL является ведущим открытым базой управления базами данных.

Все текстовые данные собирались в режиме реального времени с информационных порталов. Далее представлен алгоритм (Рисунок 1) и описание компонентов.

Первым делом подключаем нужные для парсинга библиотеки. Это Requests, BeautifulSoup4, lxml, pymusql. BeautifulSoup4 определяем, как bs. И еще нам понадобится регулярное выражение re. Через переменную headers полуачем user agent. User agent – это клиентское приложение, использующее определённый сетевой протокол. Затем используя библиотеку requests включаем сессию. И проверяем статус. После этого отправляем базовую ссылку и headers на библиотеку requests. И через библиотеку BeautifulSoup4 получаем дерево HTML. Если у сайта, который мы хотим парсит есть подстраницы, то найдя нужные тэги и его атрибуты получаем ссылки на под страницы и записываем их в .txt файл чтобы не повторять заход на сайт раз за разом, так как если клиент будет заходить на сайт многократно, то сайт заблокирует вход для безопасности. Если у сайта нет под страниц, то переходим сразу на следующий пункт. И так все нужные ссылки сохранены на .txt файл, осталось разобрать сайт на кусочки. Так как любой сайт по-своему уникален, потому что их создают разные люди, рассмотрим внимательно какие данные хранятся в каких тэгах и атрибутах.

Далее отбираем тексты в этих областях и отправляем их в нужное хранилище.

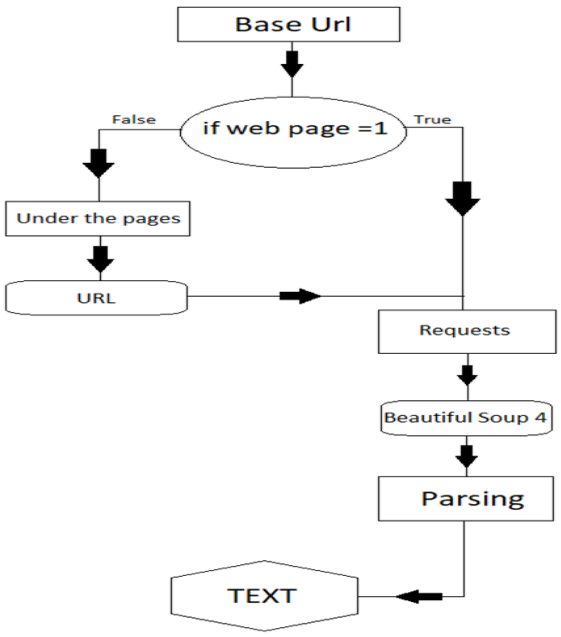

Рисунок 1. Алгоритм сбора текстовых данных в режиме реального времени

Первым делом подключаем нужные для парсинга библиотеки. Это Requests, BeautifulSoup4, lxml, pymusql. BeautifulSoup4 определяем, как bs. И еще нам понадобится регулярное выражение re. Через переменную headers получаем user agent. User agent – это клиентское приложение, использующее определённый сетевой протокол. Затем используя библиотеку requests включаем сессию. И проверяем статус. После этого отправляем базовую ссылку и headers на библиотеку requests. И через библиотеку BeautifulSoup4 получаем дерево HTML. Если у сайта, который мы хотим парсит есть подстраницы, то найдя нужные тэги и его атрибуты получаем ссылки на под страницы и записываем их в .txt файл чтобы не повторять заход на сайт раз за разом, так как если клиент будет заходить на сайт многократно, то сайт заблокирует вход для безопасности. Если у сайта нет под страниц, то переходим сразу на следующий пункт. И так все нужные ссылки сохранены на .txt файл, осталось разобрать сайт на кусочки. Так как любой сайт по-своему уникален, потому что их создают разные люди, рассмотрим внимательно какие данные хранятся в каких тэгах и атрибутах.

Далее отбираем тексты в этих областях и отправляем их в нужное хранилище.

Индексирование документов с помощью признаков

Индексирование документов с помощью признаков. Векторная модель представления текстов. Селекция признаков для классификации документов. В подавляющем большинстве систем классификации документов в качестве признаков используются слова, а под селекцией признаков понимается выделение подмножества слов, полезных для классификации, из общего словаря коллекции. Как правило, «полезные» подмножества слов формируются из ключевых (репрезентативных) слов, имеющих высокую степень близости к одному из рассматриваемых классов.

Широко распространены следующие критерии оценки близости слов к одному из рассматриваемых классов: информационная выгода, взаимная информация и критерий хи-квадрат. Указанные критерии сравниваются между собой в работе [5]. Результаты сравнения говорят в пользу критерия информационной выгоды и критерия Пирсона. Авторы работы отмечают, что взаимная информация смещена в сторону редких терминов.

Информационная выгода задается с помощью следующей формулы:

$$IG(t_k, c_i) = \sum_{c\in\{c_i,\bar{c_i}\}}\sum_{t\in\{t_k,\overline{t_k}\}} P(t,c) * log\frac{P(t,c)}{P(t)P(c)}, \quad (1)$$

где $P(t,c)$ – вероятность того, что терм t встречается во множестве документов класса c, соответственно, $P(\bar{t},c)$ - вероятность того, что терм t не встречается во множестве документов класса c.

Взаимная информация задается следующей формулой:

$$MI(t_k, c_i) = log \frac{P(t_k, c_i)}{P(t_k)P(c_i)}. \quad (2)$$

Критерий Пирсона задается следующей формулой:

$$\chi^2 = \frac{\|D\|[P(t_k,c_i)P(\overline{t_k},\overline{c_i}) - P(\overline{t_k},c_i)P(t_k,\overline{c_i})]^2}{P(t_k)P(\overline{t_k})P(c_i)P(\overline{c_i})}, \quad (3)$$

где $\|D\|$– мощность обучающей выборки.

В общем случае, селекция признаков для классификации документов представляет собой итеративный процесс. Итерации продолжаются до тех пор, пока не будет достигнуто приемлемое качество подмножества признаков. Оценка качества производится как раз на базе обучающего множества. Запускается тестовая классификация на основе обучающего множества документов, с этой целью обучающая выборка делится на 2 части: собственно, обучающую и тестовую, и оценивается качество классификации. Если качество классификации удовлетворяет каким-то пороговым значениям, подмножество признаков признается годным. На рисунке 2 представлен описанный процесс селекции признаков.

После селекции признаков, т.е. формирования признакового пространства, производится индексирование документов. Как правило, индексирование заключается в представлении документов в виде векторов, координатами которого являются измерения сформированного признакового пространства, а значениями координат – веса признаков в документе.

Рисунок 2. Типовая схема селекции признаков в задаче классификации

В работе [6] такое индексирование называется координатным. Сама модель представления текстов в виде векторов называется векторной или векторно-пространственной моделью (*VSM – vector space model*).

Пусть имеется коллекция текстовых документов $D = (d_1, d_2, \dots, d_m)$ и задан словарь терминов этой коллекции $T = (t_1, t_2, \dots, t_n)$.

Модель *VSM* представляет каждый документ этой коллекции с помощью словаря T как *n*-мерный вектор, координатами которого являются частоты вхождений терминов словаря в этот документ:

$$\bar{d}_i = (f_{i1}, f_{i2}, \dots, f_{in}), i = \overline{1,m}.$$

В некоторых модификациях модели *VSM* частоты вхождений терминов заменяют весами, которые, по сути, представляют собой те же частоты, только не абсолютные, а относительные. Существует несколько стандартных методик взвешивания терминов.

Если документы сильно различаются по длине (количеству слов), то веса терминов нормируют относительно друг друга. Без нормирования вес термина в документе будет тем меньше, чем длиннее документ. Как правило, нормирование весов осуществляют путем деления на евклидову норму (длину вектора документа):

$$\bar{d}_{norm} = \frac{\bar{d}}{|\bar{d}|} = \left(\frac{f_1}{\sqrt{{f_1}^2+\cdots+{f_n}^2}}, \frac{f_2}{\sqrt{{f_1}^2+\cdots+{f_n}^2}}, \ldots, \frac{f_n}{\sqrt{{f_1}^2+\cdots+{f_n}^2}} \right). \quad (4)$$

Особый интерес при построении векторной модели представляет формирование словаря коллекции $T = (t_1, t_2, \ldots, t_n)$. Чем больше коллекция текстов, тем выше размерность словаря. Уменьшение размерности словаря позволяет снизить вычислительную сложность алгоритмов обработки текстов. С этой целью словарь коллекции, во-первых, нормализуется, во-вторых, из него исключаются стоп-слова, в-третьих, синонимы свертываются в семантические концепты [7].

Практические результаты

Для практической реализации алгоритмов был применен комплекс языка Python. Все скаченные документаций сохранены во временное хранилище. Внутри него есть файлы для получения данных из выбранных сайтов. Рассмотрим пример с сайта akorda.kz.

Главная задача этого файла – это получение нужных сведений. Вышеупомянутых разделах все полученные ссылки храним в .txt файле. А в эти документации оно сохранено как links.txt. Для данного сайта было определено более 100 000 ссылок. Далее скаченные материалы проходили обработку: чистка от символов и тегов, убирались стоп слова, классификация документа по признакам и др. В экспериментальной работе алгоритма автоматического пополнения текстов были получены следующие результаты, представленные в таблице 1. Полученные данные, индексированные с помощью признаков, классифицированы на следующие тематики. Результаты классифицирования для некоторых тематик представлены в таблице 2.

Таблица 1. Объем обработанных данных на казахском языке

Наименование информационного портала	*Количество слов*
akorda.kz	*680 000*
nur.kz	*900 000*
wiki.py	*1.1 млрд*
ertegiler.py	*500 000*
bot.py	*2 000 000*

Таблица 2. Объем обработанных данных по тематикам

Наименование тематики	*Количество слов*
политика	*720 000*
новости	*1130 000*
спорт	*300 000*
наука	*1 100 000*
литература	*450 000*
информатика	*800 000*
техника	*500 000*

Все собранные и обработанные данные записываются в СУБД. В СУБД входят следующие основные ключевые таблицы: Title, Text, Lemmas, Segments, Genre, Keywords, Subject,Url.

По данным ключевым данных производится запрос поиска и анализ данных.

Заключение

По итогам научно исследовательской работы были получены следующие результаты: Исследованы современные системы информационных пауков; Разработан алгоритм автоматического пополнения текстов на казахском языке; Разработан алгоритм сбора текстовых данных, поступающих в режиме реального времени; Разработан алгоритм индексирования документов с помощью

признаков; Реализована программная часть алгоритмов и получены обработанные, классифицированные данные на казахском языке объемом более 1,7 млрд. слов.

Полученные научные и экспериментальные данные будут применены в исследований обработки казахского языка в интеллектуальных задач.

Исследование выполнено при поддержке Министерства образования и науки Республики Казахстан в рамках научного проекта AP 05132950.

Список использованной литературы:

1 Ломакина Л.С., Суркова А.С. Информационные технологии анализа и моделирования текстовых данных // International journal of experimental education № 12.- 2015.- С. 136-137.

2 Scrapy – Краткое руководство [Электрон.ресурс]. - URL: https: //coderlessons.com/tutorials/devops/ uchitsia-scrapy/scrapy-kratkoe-rukovodstvo (дата обращения 20.12.2019).

3 Heritrix [Электрон.ресурс]. - URL: https://en.wikipedia.org/wiki/Heritrix (дата обращения 17.01.2020)

4 Apache Nutch [Электрон.ресурс]. - URL: https://ru.qwe.wiki/wiki/Apache_Nutch (дата обращения 22.01.2020).

5 Yang Y., Pedersen J.O. A comparative study on feature selection in text categorization // Proceedings of ICML-97, 14th International Conference on Machine Learning. Morgan Kaufmann Publishers. San Francisco, US: Nashville, 1997. P. 412-420.

6 Барахнин В.Б., Ткачев Д.Н. Классификация математических документов с использованием составных ключевых терминов // Матер. Всерос. конф. с междунар. участием «Знания - Онтологии - Теории». – Новосибирск, 2009. – Т.1. – С. 16-23.

7 Разработка интеллектуальной высокопроизводительной информационно-аналитической поисковой системы обработки слабоструктурированных данных // Отчет о НИР (промежуточный) рук.: Мансурова М.Е. – Алматы, 2016. – 120 с. – № ГР 0115РК00779.

8 Романова В.О. Обзор методов средств автоматизированного сбора информации с новостных лент // Молодой ученый.-2016.-№12(116).-С.170-173.

9 Создание системы сбора и обработки открытых данных с ресурсов сети интернет [Электрон.ресурс]. - URL: http://elar.urfu.ru/bitstream/10995/61616/1/tim_2018_096.pdf (дата обращения 22.01.2020).

10 Абдуали Б.А., Әмірова Д.Т., Рахимова Д.Р., Кәрібаева А.С. Аналитическая обработка текстовых ресурсов и документов на казахском языке // Вестник КазНИТУ. – 2019. – №2 (132). – С. 356-362.

11 Shormakova A., Zhumanov Zh., Abduali B., Rakhimova D., Amirova D. Analytical Processing of Textual Resources and Documents in the Kazakh Language // Journal of Engineering and Applied Sciences. – 2019. – Vol. 14, Issue: 20. – P. 7714-7721. // DOI: 10.36478/jeasci.2019.7714.7721.

References:

1 Lomakina L.S., Surkova A.S. (2015) Informacionnye tehnologii analiza i modelirovanija tekstovyh dannyh // International journal of experimental education № 12. 136-137.

2 Scrapy – Kratkoe rukovodstvo [Jelektron.resurs]. URL: https: //coderlessons.com/tutorials/devops/ uchitsia-scrapy/scrapy-kratkoe-rukovodstvo (data obrashhenija 20.12.2019).

3 Heritrix [Jelektron.resurs]. URL: https://en.wikipedia.org/wiki/Heritrix (data obrashhenija 17.01.2020)

4 Apache Nutch [Jelektron.resurs]. URL: https://ru.qwe.wiki/wiki/Apache_Nutch (data obrashhenija 22.01.2020).

5 Yang Y., Pedersen J.O. (1997) A comparative study on feature selection in text categorization // Proceedings of ICML-97, 14th International Conference on Machine Learning. Morgan Kaufmann Publishers. San Francisco, US: Nashville, 412-420.

6 Barahnin V.B., Tkachev D.N. (2009) Klassifikacija matematicheskih dokumentov s ispol'zovaniem sostavnyh kljuchevyh terminov Mater. Vseros. konf. s mezhdunar. uchastiem «Znanija - Ontologii - Teorii». Novosibirsk,16-23.

7 Razrabotka intellektual'noj vysokoproizvoditel'noj informacionno-analiticheskoj poiskovoj sistemy obrabotki slabostrukturirovannyh dannyh Otchet o NIR(2016) (promezhutochnyj) ruk.: Mansurova M.E. Almaty, № GR 0115RK00779.

8 Romanova V.O. (2016) Obzor metodov sredstv avtomatizirovannogo sbora informacii s novostnyh lent Molodoj uchenyj.№12(116). 170-173.

9 Sozdanie sistemy sbora i obrabotki otkrytyh dannyh s resursov seti internet [Jelektron.resurs]. - URL: http://elar.urfu.ru/bitstream/10995/61616/1/tim_2018_096.pdf (data obrashhenija 22.01.2020).

10 Abduali B.A., Әmirova D.T., Rahimova D.R., Kәribaeva A.S. (2019) Analiticheskaja obrabotka tekstovyh resursov i dokumentov na kazahskom jazyke Vestnik KazNITU. №2 (132). 356-362.

11 Shormakova A., Zhumanov Zh., Abduali B., Rakhimova D., Amirova D.(2019) Analytical Processing of Textual Resources and Documents in the Kazakh Language Journal of Engineering and Applied Sciences 7714-7721. DOI: 10.36478/jeasci.2019.7714.7721.