УДК 681.3: 658.56

А.Н. Глибовец, И.В. Решетнев

Программный модуль автоматизированного построения тезаурусов в формате *RDF* из украиноязычных текстов формата *pdf*

Описана разработка программного модуля в виде веб-сервиса для автоматизированного построения тезаурусов на основе комбинированного метода выявления важных терминов и связей в тексте и алгоритма автоматизированного итеративного построения терминологий в коллекциях научных текстов на украинском языке. Формат тезауруса *JSON-LD* избран с учетом возможности публикации полученных терминологических связей в стандартизированном виде сетевого доступа к ресурсам, и с позиций понимания тезауруса как полноценного программного модуля поисковой системы научных материалов.

The development of a software module in the form of a web service for thesauruses automated construction based on the combined method of identifying important terms and links in the text and automated iterative algorithm for constructing terminologies in the collections of scientific texts in Ukrainian language is described. Thesaurus format JSON-LD was elected with the possibility to publish the terminological relationships in a standardized form of network access to the resources, and from the standpoint of understanding the thesaurus software module as a part of search engine system.

Описано розробку програмного модуля у вигляді веб-сервісу для автоматизованої побудови тезаурусів на основі комбінованого методу виявлення важливих термінів і зв'язків у тексті і алгоритму автоматизованої ітеративної побудови термінологій в колекціях наукових текстів українською мовою. Формат тезауруса *JSON-LD* обрано з урахуванням можливості публікації отриманих термінологічних зв'язків у стандартизованому вигляді мережевого доступу до ресурсів, і з позицій розуміння тезауруса як повноцінного програмного модуля пошукової системи наукових матеріалів.

Введение. Сфера научных исследований – наиболее продуктивная форма как обогащения человеческого понятийного пространства новыми концептами и связями, так и непосредственно связанным с этим лексическим процессом словообразования для описания новых концепций. Зачастую в таких условиях темпы создания и актуализации словарей не поспевают за прогрессом в новейших исследованиях, в силу объективных причин сложности изучаемых сфер и изменчивости понятий со временем [1]. Вместе с тем сохраняется острая необходимость взаимопонимания среди исследователей, что требует как унифицированной и доступной терминологической базы, так и качественной поисковой системы научных документов.

Одним из эффективных способов улучшения релевантности поисковой выдачи таких систем служит использование тезауруса, т.е. справочника характера и силы связей между терминами. Автоматизированный метод построения тезаурусов лучше других подходит для сферы научных исследований из-за высоких темпов обновления информации и связанной высокой себестоимостью участия экспертов. В [2] описан разработанный авторами метод автоматизированного определения важных украиноязычных терминов и терминологических связей между ними и *алгоритм автоматизированного итеративного построения* терминологий в коллекциях научных текстов на украинском языке. Также описана разработка программной реализации предложенного подхода как компонента поисковой системы украиноязычных научных документов, созданной на факультете информатики Национального университета «Киево-Могилянская академия».

Актуальность данного исследования обусловлена появлением в открытом доступе прикладных решений для анализа текстов на украинском языке с возможностями использова-

Ключевые слова: *Google API*, *MongoDB*, *Apache Lucene*, *PDFBox*, *Jlemmagen*, *JlanguageTool*, *Spring REST*, документ, связь «общее–частное», гипонимия.

ния таких методов, как лемматизация и тегирование по частям речи. При разработке модуля учтены ограниченность выпущенных документарных коллекций на украинском языке и возможность итеративного добавления документов в терминологическую базу с последующим обновлением содержания тезауруса.

Программная реализация

Сбор данных. Этап начального сбора данных необходим для построения справочной системы документарных частот терминов, а также для использования при проведении тестирования алгоритма на точность составления тезаурусов. Поэтому для сбора документов из открытых источников, в частности выпусков журнала «Научные записки НаУКМА», был разработан специальный скрипт, который перебирал страницы сайтов архива [3] и сохранял все документы в каталогах по годам и темам. Разбиение по темам использовано для создания различных тематических тезаурусов на этапе тестирования. Всего из данной коллекции было использовано 2926 текстовых документов за 1996–2013 годы издания.

Справочный компонент документарной частоты. ***Решение на основе внешней поисковой системы.*** Построить документарную частоту по небольшой коллекции, а тем более по одному документу очень сложно. Проще использовать уже существующую документарную частоту термина из поисковой системы. В качестве примера была выбрана поисковая система *Google*, которая вместе с результатами обычного поиска всегда возвращает приблизительное количество результатов поисковой выдачи. Данный показатель и был взят в качестве документарной частоты термина. Во время разработки вспомогательных методов для извлечения данного показателя из поисковой выдачи было принято решение, что обычный парсинг веб-страниц поисковой выдачи – это громоздкое решение, так как для одного числовго показателя приходилось запрашивать и разбирать целую веб-страницу, содержащую много несущественных данных. Поэтому был написан клиент сервиса *Google Custom Search RESTful API* [4] с соответствующими настройками в консоли сервиса *Google*. В качестве результата был реализован программный метод, возвращающий по данному термину его документарную частоту, причем количество интернет-трафика для таких запросов минимален.

Однако возникает другая проблема для использования *Google API*: квоты свободного доступа, которые составляют 100 запросов в день. Конечно, чтобы обрабатывать большие документы и определять, какие термины в них важны, необходимо узнать документарную частоту каждого. Поэтому было принято решение, во-первых, хранить в локальном кэше все найденные документарные частоты, и в первую очередь обращаться к нему, что с течением времени приведет к образованию достаточно полной базы терминов, и, во-вторых, сохранить возможность обрабатывать обычные веб-страницы поиска на случай исчерпания квоты.

Кэш, как и другие данные промежуточной работы алгоритма, было решено хранить в документарной базе *MongoDB*.

Решение на основе индексации эталонной коллекции документов. В качестве эталонной коллекции выбраны все выпуски журнала «Научные записки НаУКМА», полученные на этапе сбора данных статей в открытом доступе.

В процессе подготовки решения структура каталогов эталонной коллекции была выровнена, и все документы собраны в одной директории. Для осуществления индексации коллекции использовано решение от *Apache Lucene*, так же, как и в финальном варианте рабочей системы. Доступ к программному интерфейсу библиотеки индексации осуществлялся из *Java*-кода, и специально для данной задачи создания начального индекса документарных частот терминов была создана отдельная конфигурация запуска программы. Этапы работы данной программы таковы:

- Считывание параметров директории, содержащей документы коллекции, а также отдельной временной конфигурации документарной базы данных *MongoDB* для хранения промежуточных результатов поиска документарных частот.

• Создание нового индекса *Apache Lucene*.

• Подокументный разбор *PDF* и добавление текстов в индекс.

• Просмотр индекса и сохранение документарных частот всех терминов в документарной базе.

• Дамп документарной базы для дальнейшего использования в качестве начального наполнения коллекции документарных частот при обычной работе метода.

В итоге после индексации коллекции и фильтрации терминов в дампе насчитывалось около 143 тыс. терминов.

Выбор готовых решений для реализации алгоритма. При разработке метода итеративного построения терминологии были использованы готовые решения для реализации элементарных операций статистических и лексикографических методов, таких как разбор документов формата *pdf*, индексирование, подсчет частот терминов, поиск документов в коллекции по фразе и тегирование по частям речи. Рассмотрим выбранные авторами инструменты для решения данных прикладных задач.

Библиотеки и утилиты *Apache Lucene* [5] использованы в качестве основы для системы индексации входящих текстовых документов и разбиения на однословные термины, а также в качестве базовой системы поиска характеристических фрагментов текста, что на основе построенного индекса позволяет найти документы, в которые входит искомый термин, и таким образом значительно ускоряет поиск в сравнении с линейным прохождением по всем документам.

PDFBox [6] – утилитарная библиотека для разбора файлов в формате *pdf*. Главное преимущество данного решения – совместимость с форматом документа *Apache Lucene*, что позволяет передавать разобранный документ сразу в подсистему индексации.

Jlemmagen [7] – один из ключевых компонентов готовых решений для разбора украиноязычной терминологии, представляющий собой коллекцию утилит для работы в сфере обработки естественного языка, в частности содержащий лемматизатор, способный приводить поданные на вход слова из текста в нормальную форму в соответствии с правилами языка. Данное решение достаточно ново, и его основное преимущество – поддержка украинского языка, а также совместимость с языковыми анализаторами библиотек индексации *Apache Lucene*.

JlanguageTool [8] – библиотека с набором утилитарных методов работы с текстом, содержащая компонент тегирования по частям речи, поддерживающая украинский язык, приводящая слова в нормальную форму, а также синтезирующая словоформы по указанным тегам. Это решение использовано в первую очередь для реализации лексикографических методов сопоставления с шаблонами.

Google Custom Search API [9] – сервис, использованный для одного из способов получения репрезентативной документарной частоты термина из внешней поисковой системы.

MongoDB [10] – документарная база данных, использованная в качестве основного хранилища разработанной системы. В базе хранятся как собственно термины с посчитанными документарными частотами, так и связи между терминами, составляющими информационную основу тезуаруса. Выбор документарной базы обусловлен совместимостью форматов: принятое решение спроектировано для поддержки формата *JSON*, что в свою очередь служит базой для формата *JSON-LD* как конкретной спецификации *RDF*, выбранной для публикации тезауруса. Таким образом, с архитектурной точки зрения, в системе будет представлен только один формат данных как для хранения, так и для публикации данных клиентам через веб-сервисы, что должно обеспечить масштабируемость и поддержку такой системы в дальнейшем.

Spring REST [11] – *java*-фреймворк, предназначенный для поддержки разработки программных систем в виде *RESTful* веб-сервисов. Использован в качестве базовой технологии при разработке программного прикладного интерфейса, что предоставляет доступ к скачиванию готового тезауруса, добавлению новых

документов в коллекции и навигации по терминам и связям различных тезаурусов.

Архитектура системы обработки документов для построения терминологии. В качестве системы, которая должна предоставить удобный доступ к функциональности разработанного метода, было решено разработать веб-сервис с возможностью построения из имеющихся у пользователя документов тезауруса в формате *RDF* (рис. 1).

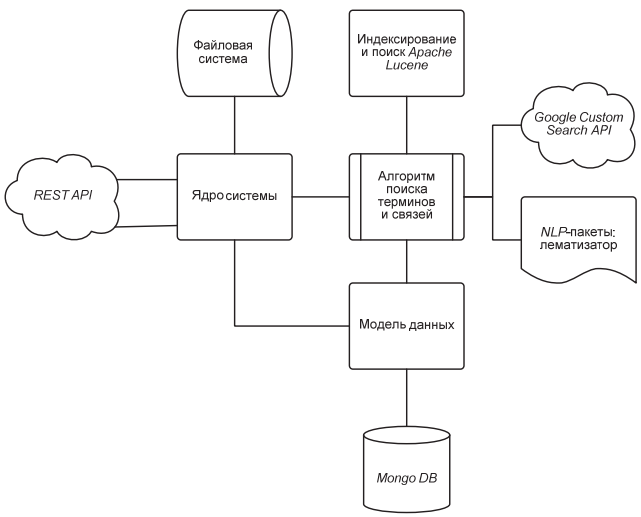

Рис. 1. Компонентная схема системы

Согласно приведенной схеме, роль доступа к функциональности ядра системы отведена на уровень *REST*-контроллеров системы. Функции ядра состоят в обработке запросов от программного интерфейса и в интеграции разработанных компонентов системы и алгоритмов. В частности, ядро имеет доступ к файловой системе для сохранения полученных при загрузке в коллекцию документов, а также через промежуточную модель данных получает доступ к документарной базе во время сериализации тезауруса в формате *RDF*. К алгоритмам поиска терминов и связей, в свою очередь тесно связанным с подсистемами индексации документов и вспомогательными библиотеками тегирования по частям речи, относятся статистические методы поиска необходимых терминов на основе взвешивания, методы поиска связей по лексикографическим шаблонам и вспомогательные подходы, связанные с доступом к данным, поиском характеристических фрагментов текста, согласованием словосочетаний.

Более детально реализацию компонентов системы можно представить в виде диаграммы классов (рис. 2).

Классы *IndexController* и *RDFController* ответственны за предоставление HTTP-интерфейса веб-сервиса к функциям управления коллекциями, добавления новых документов и просмотра тезаурусов в нескольких вариантах форматирования.

Группа классов *Termin*, *TermInDoc* и *TermRelation* относится к модели данных и используется как на этапе применения алгоритма поиска, так и в качестве спецификации структуры соответствующих сущностей документарной базы *MongoDB*. С помощью абстракции репозитория, предоставляемой фреймворком *Spring Data* [13], можно применить статическую типизацию интерфейсов соответствующих *TerminRepository*, *TermInDocRepository* и *TermRelationRepository* для связи приведенных прокси-классов с библиотеками работы с базой данных.

Класс *IndexFacade* выполняет роль ядра системы, сочетая компоненты системы в определенные функциональные блоки для выполнения основных операции, таких как добавление и индексация нового документа.

Среди классов, непосредственно содержащих методы поиска терминов и связей, ключевая роль принадлежит *TopTfIDfNounExtractor* и *RelationFinder*. *Первый* выполняет инструкции статистического этапа разработанного метода по поиску необходимых терминов, их взвешиванию, сортировке и фильтрации полученного их списка. *Второй* непосредственно содержит статически заданную коллекцию лексикографических шаблонов, и получает с их помощью связи между терминами, поочередно применяя шаблоны к коллекциям предложенных характеристических фрагментов текста.

Кроме приведенных на диаграмме классов, в коде реализации существуют и другие утилитарные компоненты системы, связанные с интеграцией с готовыми решениями и

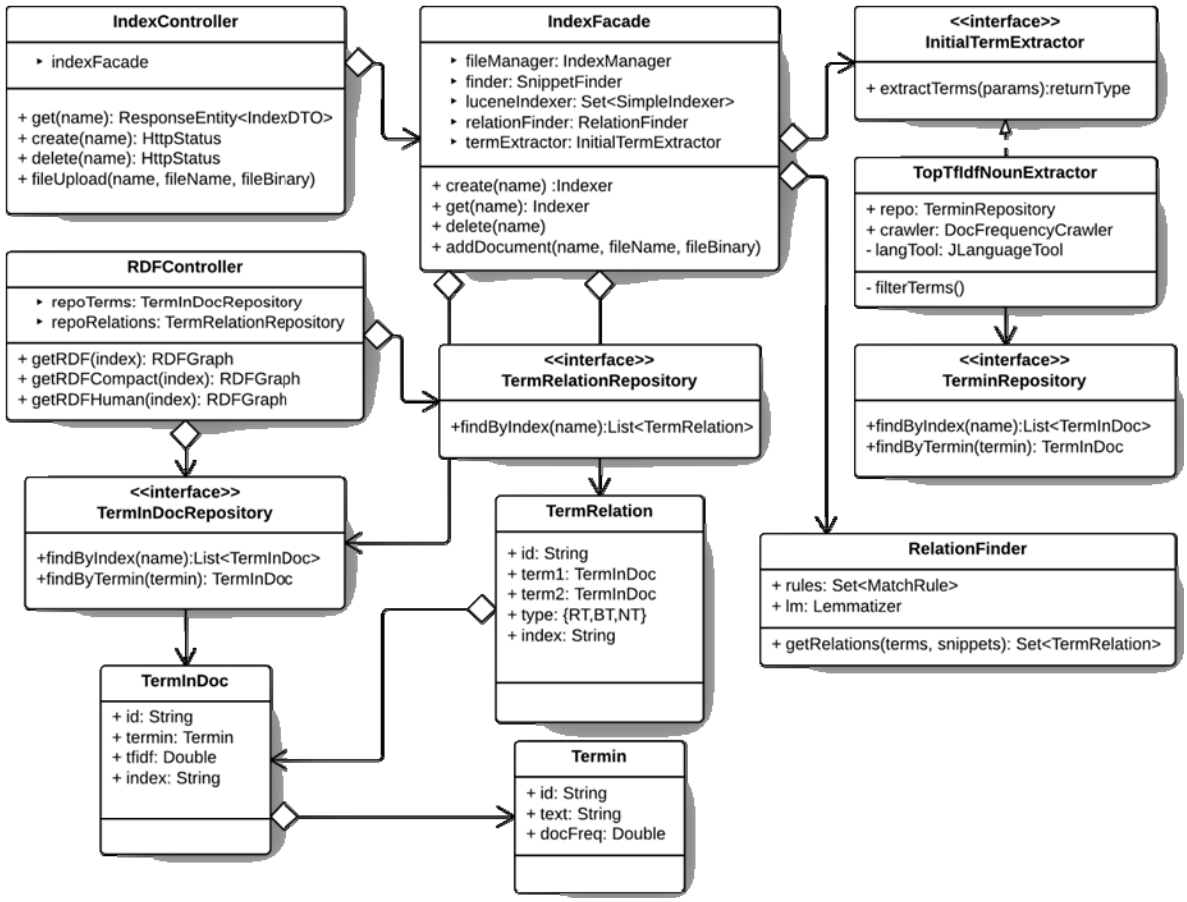

Рис. 2 *UML*-диаграмма основных классов системы построения тезаурусов

решением прикладных задач по обработке текста и управлению коллекциями на уровне пользователя.

Спецификация программного интерфейса доступа к системе. Прикладной программный интерфейс разработанной системы представляет собой способ доступа к программе с возможностями создания и просмотра тезаурусов. Предусмотрены два варианта доступа к функциональности системы: *RESTful API*, что позволяет обращаться к системе по протоколу *HTTP*; консольное приложение для локальной пакетной обработки коллекций документов, используемое в частности для тестирования алгоритма.

Формат сериализации тезауруса на базе *JSON-LD*. К основным секциям данного формата относится узел ***@context***, в котором приводится список ссылок к типам данных внутри тела документа, и собственно начальный узел тела документа, обозначенный как ***@graph***. В рамках работы над форматом представления тезаурусов проведен анализ формата *JSON-LD*, в особенности рекомендаций относительно обозначения абстрактных концептов *RDF* для тезаурусов в соответствии со стандартом *ISO*-25964 [12]. В результате определен минимальный набор полей и их типов, удовлетворяющих потребности сериализации терминов и связей. Ссылки на соответствующие типы данных указаны в узле контекста.

Таблица 1. Спецификация точки доступа программного интерфейса работы с тезаурусом

***HTTP*-метод**	***URI***	**Предназначение**
GET	*/rdf/{indexName}/*	Получить тезаурус по имени
GET	*/rdf/{index Name}*/compact	Доступ к тезаурусу в компактной форме. В список терминов включены только те, для которых найдены связи
GET	/rdf/{*index Name*}/human	Вывод тезауруса в удобной форме для чтения и просмотра связей

Таблица 2. Спецификация точки доступа к программному интерфейсу управления коллекциями

HTTP-метод	*URI*	Предназначение
GET	/index/	Получить список коллекций документов в системе
POST	/index/{*indexName*}/	Создание новой коллекции с именем
GET	/index/{*indexName*}/	Просмотр статистической информации по коллекции: название, количество документов, терминов и найденных связей
DELETE	/index/{*indexName*}	Удаление коллекции и связанного тезауруса
POST	/index/{*indexName*}/upload	Загрузка файла в формате *pdf* к коллекции документов; метод инициирует запуск алгоритма перестроения тезауруса с учетом данных нового документа

Пример сериализованного в формате *JSON-LD* тезауруса приведен далее.

```
{
    "@context" : {
      "iso25964" : "http://www.niso.org/schemas/iso25964/iso25964-1_v1.4.xsd#",
      "thesaurus" : "iso25964:Thesaurus",
      "concept" : "iso25964:ThesaurusConcept",
      "relation" : "iso25964:HierarchicalRelationship",
      "role" : "iso25964:role",
      "baseConcept" : "iso25964:isHierRelConcept",
      "depConcept" : "iso25964:hasHierRelConcept",
      "lexicalValue" : "iso25964:lexicalValue"
    },
    "@graph" : {
      "thesaurus" : {
        "concept" : [
                {
                  "@id" : "C1",
                  "lexicalValue" : "наука"
                }, {
                  "@id" : "C2",
                  "lexicalValue" : "філософія"
                }
        ],
        "relation" : [
        {
              "baseConcept" : "C1",
              "depConcept" : "C2",
              "role" : "NT"
            }
        ]
}
```

Непосредственно к узлу тезауруса в теле документа, в соответствии с указанным форматом, размещаются коллекции концептов и связей. При этом структура концептов связывает их идентификаторы с лексическими значениями терминов, а связи в свою очередь связывают идентификаторы терминов между собой с указанием типа связи.

Анализ результатов

Приведены результаты тестирования и применения алгоритма и модуля для построения терминологии в виде *RDF*-схемы с использованием реальных текстовых коллекций украиноязычной научной периодики.

Схема тестирования и оценка результатов. Для тестирования работы алгоритма было решено разработать систему конфигураций для удобного и гибкого просмотра различных вариаций алгоритма на различных данных с учетом необходимости оптимизации скорости таких тестирований, а также с замером основных характеристик выполнения отдельных шагов алгоритма.

Для данного случая наиболее приемлема реализация отдельных интеграционных тестов, которые будут содержать независимые конфигурации окружения, такие как реализация документарной базы в памяти для каждого теста [13].

Схема тестирования – необходимый компонент системы как для исследования точности найденных терминологических связей между терминами, так и для рассмотрения модификаций алгоритма при поиске наиболее эффективного подбора параметров, включаемых в финальную реализацию системы как веб-сервиса.

При создании схемы тестирования проведены следующие работы: получен единый формат тезауруса для сравнения; зафиксирована модификация разработанного метода; настроена тестовая среда, позволяющая запускать просмотры алгоритмов; найдена совокупность реальных текстовых научных материалов для сравнения алгоритмов; разработана метрика точности поиска.

Для проверки результатов работы алгоритма было решено построить несколько тезаурусов на основе различных тематических разделов украиноязычного журнала «Научные записки НаУКМА», сгруппировав коллекции текстов по различным темам за несколько лет, а также двух контрольных тезаурусов, не связанных с данным журналом. Контрольные тезаурусы были составлены документами из периодических изданий на две темы, не включенные в эталонную коллекцию. Такой контроль необходим для минимизации влияния

начального фильтра необходимых терминов, который в данной статье построен на основе всех тем журнала «Научные записки НаУКМА» за все годы, и поэтому заведомо лучше работающем, чем с произвольной коллекцией.

В сводную таблицу по каждому тезаурусу собрана следующая статистика:

- количество: документов в базовой коллекции текстов;
  - найденных терминов при начальной фильтрации;
  - добавленных терминологических словосочетаний и общее количество терминов;
  - найденных связей;
  - связей по типам *RT*, *BT* и *NT*;
- измеренный коэффициент точности поиска связей.

Коэффициент точности метода измерялся следующим образом. Из составленного тезауруса избирались случайно N найденных связей и проводилась оценка релевантности связи между терминами на основе обращения к первоначальному тексту.

Коэффициент точности имел следующее правило расчета:

$$c_{pr} = \frac{2N1 + 0,5 * N2}{2,5N},$$

где: $N1$ соответствует количеству релевантных связей из N без учета направления связи; $N2$ – количество правильно распознанных по направлению связей типа *BT* и *NT* соответственно.

Введенные в формулу расчета весовые коэффициенты выделяют первостепенную важность нахождения простых связей.

Результаты тестирования метода на тематических коллекциях документов. В среднем точность разработанного метода на данных тестовых запусках составляет 70,5 процента, и такой показатель приемлем для данного метода.

В результате тестирования на примерах коллекций замечены следующие закономерности:

- большинство совпадений приходится на шаблон *LP*1 [2], обозначающий прямые определения в тексте;
- метод предоставил, как и было предусмотрено, меньшую точность поиска на контрольных коллекциях, однако деградация точности не стала существенной;
- шаблоны типа *LP*2-4 [2] находили довольно мало связей, около 20 на 6 тыс. фраз.

Отмеченные закономерности свидетельствуют о том, что лексикографический метод довольно чувствителен к формальной записи отдельных шаблонов, а также указывает на недостаточную полноту коллекции по размеру и покрытию научной сферы. Действительно, коллекции из 50–90 документов разносторонних статей, объединенных только широкой тематикой научной отрасли, публикуемых в журнале «Научные записки НаУКМА» и в других периодических изданиях, нельзя считать достаточно полными, чтобы метод лексикографических шаблонов по Хеарсту дал ожидаемые результаты.

На основе анализа можно внести следующие предложения относительно увеличения полноты и точности метода: использование стохастического метода для устранения неоднозначно-

Таблица 3. Результаты тестирования метода на коллекциях научных текстов

№	Название коллекции	Количество документов	Найдено терминов	Построено связей		Релевантность			Оценка точности, C_{pr}
				RT	*BT*, *NT*	*N*	*N*1	*N*2	
1	Компьютерные науки	96	785	47	195	20	18	14	0,86
2	Философия и религиоведение	45	1026	35	234	20	16	12	0,76
3	Экономические науки	58	890	20	195	20	15	15	0,75
4	Юридические науки	66	748	51	47	20	13	10	0,62
5	Биотехнологии, контрольная коллекция [1]	35	464	10	142	20	12	11	0,59
6	Социология, контрольная коллекция [8]	27	627	28	94	20	14	9	0,65

стей в трактовке тегов частей речи слов в контексте; использование большего количества грамматических правил согласования слов в терминологических словосочетаниях во время приведения фраз к нормальной форме; увеличение количества лексикографических шаблонов и длины синонимических рядов определяющих лексем шаблона для достижения большей полноты поиска связей; применение больших коллекций документов для анализа, а также для построения эталонной коллекции сроков документарных частот.

Заключение. Тестирование реализации предложенного метода на тематических коллекциях научных текстов продемонстрировало эффективность алгоритма и достаточную его точность в рамках разработанных шаблонов. Полученный программный модуль продемонстрировал свои прикладные возможности для тестовых коллекций данных, и может быть использован как составляющая часть поисковой системы украиноязычных научных материалов.

1. *Искусство* и ремесло лексикографии Сидни И. Лендау: Словари. – К.: К.И.С. – 2012. – 480 с.
2. *Алгоритми* обробки текстів вільної форми для отримання фактів і зв'язків між ними / А.М. Глибовець, О.О. Марченко, Д.В. Циганок та ін. // Наук. записки НаУКМА. Комп'ютерні науки. – Т. 138. – 2012. – С. 35–39.
3. *Наукові* записки НаУКМА: Архив. Веб. 12.04.2014 – <http://nz.ukma.edu.ua/index.php?option=com_content-&task=section&id=10&Itemid=47> і <http://www.ekmair.ukma.kiev.ua/>
4. *Custom* Search // Google Developers. Web. 08 Apr. 2014. – <https://developers.google.com/custom-search/json-api/v1/overview>
5. *Apache* Lucene Core // Apache Lucene. Web. 08 Apr. 2014. – <http://lucene.apache.org/core/>
6. *Apache* PDFBox – A Java PDF Library // Apache PDFBox. Web. 08 June 2014. – <http://pdfbox. apache. org/>
7. *Bitbucket*. Hlavki // JLemmaGen. Web. 08 Apr. 2014. – <https://bitbucket.org/hlavki/jlemmagen>
8. *Language*Tool Wiki. – Java API. Web. 08 Apr. 2014. – <http://wiki.languagetool.org/java-api>
9. *Custom* Search // Google Developers. Web. 08 Apr. 2014. – <https://developers.google.com/custom-search/json-api/v1/overview>
10. *Agile* and Scalable // MongoDB. Web. 05 Apr. 2014. – <http://www.mongodb.org/>
11. *Building* a RESTful. Web Service // Getting Started. Web. 08 Apr. 2014. – <https://spring.io/guides/gs/rest-service/>
12. *ISO* 25964 – the International Standard for Thesauri and Interoperability with Other Vocabularies // ISO 25964 Thesaurus Schemas. Web. 08 Apr. 2014. – <http://www.niso.org/schemas/iso25964/>
13. *Spring* MongoDB Tutorial // Spring MongoDB Tutorial. Web. 08 Apr. 2014. – <http://bits-and-kites.blogspot.com/2014/01/spring-mongodb-tutorial.html>

Поступила 18.07.2014
Тел. для справок: +38 067 409-4355, 095 506-6611 (Киев)
E-mail: *andriy@glybovets.com.ua*, *reshet.ukr@gmail.com*