Автоматическая система классификации текстов для базы знаний предприятия

Попков М.И.

***Аннотация*— В работе рассмотрены методы машинного обучения для решения задачи классификации данных. Проведено исследование методов индексации, взвешивания и классификации для корпуса документов базы знаний предприятия. Рассмотрены метрики сравнения классификаторов и получены результаты сравнений в рамках существующей инфраструктуры.**

Предложен способ использования существующего поискового индекса для решения задачи классификации документов. Разработана реализация и сценарии работы автоматической системы классификации текстов.

***Ключевые слова*—Классификация текстов, поисковой индекс, веб-сервис.**

I. Введение

В связи с научно-техническим прогрессом объем документации на предприятиях имеет тенденцию к постоянному и все ускоряющемуся увеличению. Согласно результатам исследования компании IDC в работе Digital Universe Study, объем накопленных данных в компаниях будет удваиваться каждые 18 месяцев (утверждение на 2009ый год). [1]

Согласно исследованиям компании Docflow [2], на поиск необходимой информации и документов уходит более 9 часов в неделю. Поиск является одним из проблемных мест в управлении корпоративными данными. Для успешного ведения бизнеса компаниями необходимы автоматические системы поиска и анализа информации.

Целью данной работы является создание автоматической системы классификации текстов базы знаний. Рассмотрены инструменты для создания автоматической системы и способы её интеграции в существующую инфраструктуру.

II. Начальные сведения и определения

A. Описание исходных данных

Под электронной базой знаний предприятия будем понимать совокупность программных средств, обеспечивающих поиск, хранение, преобразование и запись сложно структурированных информационных единиц (знаний). [3]

Статья получена 2 июня 2014.
Попков Максим Иванович email: maxim.popkov@outlook.com
Магистрант факультета ВМК МГУ имени М.В. Ломоносова.
Работа представляет результаты магистерской диссертации.

В рамках данной работы, рассматривается база знаний небольшого предприятия. Типичные виды документов базы знаний рассматриваемого предприятия – это тексты по информационным технологиям, научные статьи, описания алгоритмов и презентации с конференций.

Характерные особенности документов и категорий рассматриваемой базы знаний:

- Количество слов в документе от 350 до 7000.
- Количество категорий от 4 до 10.
- Число документов внутри категории от 2 до 60.
- Категории тематически близки.
- Между категориями нет иерархической зависимости.
- Категории организованы в виде плоской структуры.

B. Постановка задачи

Для существующей базы знаний предприятия необходимо:

• Предложить методы для индексации и автоматической классификации данных.

• Разработать классификатор удовлетворяющий требованиям предприятия.

В рассматриваемом предприятия существует несколько баз знаний, поэтому разрабатываемый классификатор данных должен быть внешним по отношению к разрабатываемой системе и не зависеть от её технических особенностей.

Предполагается что количество документов в системе относительно мало и не превышает 10 000 документов.

C. Задача классификации

Пусть:

- $D = \{d_1, \dots, d_{|D|}\}$ — множество документов,
- $C = \{c_1, \dots, c_{|C|}\}$ — множество категорий,
- $\Phi: D \times C \to \{0, 1\}$ — неизвестная целевая функция, которая по паре $< d_i, c_j >$ говорит, принадлежит ли документ d_i категории c_j (1 или T) или нет (0 или F). [4]

Задача классификации состоит в построении классификатора $\Phi': D \times C \to \{0, 1\}$, максимально близкого к Φ.

Выше была поставлена задача точной классификации, т.е. каждый документ относится только к одной категории. В работе используются классификаторы с ранжированием, при котором множество значений целевой функции – это значения из интервала [0,1]. Документ при ранжировании может относится сразу к

нескольким категориям с разной степенью принадлежности.

Методы машинного обучения, используемые для классификации, полагаются на наличие коллекции $\Omega = \{d_1,\ldots,d_{|\Omega|}\}$, $\Omega \subset D$ заранее классифицированных документов, то есть таких, для которых точно известно значение целевой функции Ф. Для того, чтобы после построения классификатора можно было оценить его эффективность, Ω разбивается на две части, не обязательно равного размера. [4]

- Обучающая (training-and-validation) коллекция. Классификатор строится на основании характеристик этих документов.
- Тестовая (test) коллекция. На ней проверяется качество классификации.

D. Этапы классификации

Классическая задача классификации может быть разбита на два основных этапа:

- Предобработка/Индексация – отображение текста документа на его логическое представление, например, вектор весов $\overline{d_j}$, который затем подается на вход алгоритму классификации.
- Классификация/Обучение – этап классификации документа или обучения на множестве документов, основанный на логическом представление документа. Важно отметить, что для классификации и обучения может быть использован общий метод предобработки/индексации текстов.

Этап предобработки отображает текст документа на логическое представление.

Текст представляется в виде мультимножества термов (слов). Множество всех термов $T = \{\tau_0,\ldots,\tau_{|T|}\}$. Каждому терму $\tau_i \in T$ сопоставлен некоторый вес w_{ij}, $0 \le w_{ij} \le 1$, числовая характеристика встречаемости этого слова в документе $d_j \in D$.

Логическое представление документа $\overline{d_j}$ - вектор n-мерной размерности $\overline{d_j} = \langle w_{1j},\ldots,w_{|T|j} \rangle$. где каждый компонент w_{ij} является весом i-го терма из множества термов T в документе $\overline{d_j}$. Полученное в итоге n-мерное пространство векторов принято называть *пространством признаков* для категории документов D. Каждый документ - это точка в пространстве признаков.

Размерность вектора — это количество термов, которые встречаются в документах множества D.

Предобработка документа, в таком случае, это преобразование последовательности термов документа в n-мерное векторное пространство.

Процесс получения вектора весов $\overline{d_j}$ для документа называют индексацией документа. Индексацию можно представить в виде трех этапов:

- Извлечение термов *(Term extraction)* – на этом этапе применяются методы для поиска и выбора наиболее значимых термов в корпусе документов.
- Взвешивание термов *(Term weighting)* – определение значимости терма для выбранного документа.
- Уменьшение размерности векторов (Dimensionality reduction) – процесс сокращения векторного пространства.

E. Взвешивание термов с использованием статистических мер

TF ***(term frequency — частота терма)*** — отношение числа вхождения некоторого терма к общему количеству термов документа. Таким образом, оценивается важность терма τ_i в пределах отдельного документа d_j. [6]

Пусть fr_{ij} — число вхождений терма τ_i в документ d_j. Тогда частота терма определяется как:

$$TF(\tau_i, d_j) = \frac{fr_{ij}}{\sum_i fr_{ij}} \quad (1), \text{ где } 0 \le i \le |T|, 0 \le j \le |D|$$

IDF ***(inverse document frequency — обратная частота документа)*** — инверсия частоты, с которой некоторое слово встречается в документах коллекции. Учёт IDF уменьшает вес широкоупотребительных слов. Для каждого уникального слова в пределах конкретной коллекции документов существует только одно значение IDF. [6]

$$IDF(\tau_i, D) = \frac{|D|}{|(d_i \supset t_i)|} \quad (2),$$

где, |D| — количество документов в коллекции, $|(d_i \supset \tau_i)|$ — количество документов, в которых встречается τ_i (когда $fr_i \neq 0$), $0 \le i \le |T|$.

TF-IDF — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален количеству употребления этого слова в документе и обратно пропорционален частоте употребления слова в других документах коллекции. [5, 6] Вес терма τ_i в документе d_j вычисляется как:

$$w_{ij} = TF(\tau_i, d_j) * IDF(\tau_i, D) \quad (3)$$

TF-SLF

Мера TF-IDF рассматривает важность терма в рамках всего корпуса документов. При такой оценке игнорируется важность терма в рамках отдельно взятой категории. Предположение что оценка терма не должна зависеть от категории и должна быть одинаковой в рамках всего корпуса может работать не эффективно, когда документов в корпусе меньше 1000 и они являются тематически близкими. Это было показано в работе [7] и подобное ухудшение качества классификации наблюдается для корпуса документов рассматриваемой базы знаний.

Для преодоления данного ограничения, рассмотрим метрику TF-SLF [7] основанную на следующих предположениях:

- Терм является важным в рамках категории, если он встречается в большинстве документов данной категории.
- Оценка терма понижается, если он является важным для нескольких категорий.

Введем следующие обозначения:

1. NDF_{tc} – нормализованная частота встречаемости терма t в категории c,
2. df_{tc} – число документов категории c в которых встречается хотя бы раз терм t,
3. N_c – количество документов в категории c,
4. C – множество категорий в корпусе документов,

5. SLF_t – логарифмированная сумма частот терма *t*.

$$NDF_{tc} = \frac{df_{tc}}{N_c} \quad (4)$$

Оценка NDF_{tc} локальна для категории. Для получения глобальной оценки R_t в рамках всего корпуса все NDF_{tc} суммируются:

$$R_t = \sum_{c \in C} NDF_{tc} \quad (5)$$

Логарифмированная сумма частот вычисляется как:

$$SLF_t = \log \frac{|C|}{R_t} \quad (6)$$

SLF позволяет устранить дисбаланс между категориями с малым числом документов и категориями с большим числом документов. [15]

Оценка TF-SLF для терма *t* вычисляется как:

$$TFSLF_t = TF_t * SLF_t \quad (7)$$

Реализация алгоритмов TF, TF-IDF уже существует в популярных библиотеках для работы с текстами.

Реализация алгоритма TF-SLF для данной работы написана на языке python. Для ускорения работы с большими матрицами термов используется библиотека numpy. Основные вычисления приведены к матричному виду.

Обозначения:

1. Число категорий: **$|C| = m$.**
2. Число термов в пространстве признаков: **$|T| = n$.**
3. ***DF*** – матрица, элементами которой являются df_{ij} , строки – категории, столбцы – термы.
4. ***NC*** – диагональная матрица, каждая строка соответствует категории, каждый элемент диагонали равен количеству документов в соответствующей категории в минус первой степени.
5. ***R*** – вектор столбец, каждый элемент равен сумме локальных частот терма.

Для вычисления метрики *TF-SLF* необходимо:

1. Составить матрицу *DF* частот совместного вхождения в категорию терма и документа, и диагональную матрицу *NC*.

$$DF = \begin{pmatrix} df_{11} & \cdots & df_{1n} \\ \vdots & \ddots & \vdots \\ df_{m1} & \cdots & df_{mn} \end{pmatrix} \quad (8)$$

$$NC = \begin{pmatrix} 1/N_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/N_m \end{pmatrix} \quad (9)$$

2. Получить нормализованную матрицу локальных частот *NDF:*

$$NDF = NC * DF \quad (10)$$

3. Рассчитать вектор-столбец глобальных оценок частот по каждому терму:

$$R = \begin{Bmatrix} ndf_{11} + ndf_{21} + .. + ndf_{m1} \\ \dots \\ ndf_{1n} + ndf_{2n} + .. + ndf_{mn} \end{Bmatrix} = \begin{pmatrix} R_1 \\ \vdots \\ R_n \end{pmatrix} \quad (11)$$

4. И получить вектор-столбец SLF:

$$SLF = \begin{Bmatrix} log \frac{|C|}{R_1} \\ \dots \\ log \frac{|C|}{R_n} \end{Bmatrix} = \begin{pmatrix} SLF_1 \\ \vdots \\ SLF_n \end{pmatrix} \quad (12)$$

Для расчета матрицы TFSLF необходимо вектор-столбец транспонировать и умножить поэлементно на каждую строку матрицы TF:

$$TFSLF = \begin{pmatrix} TFSLF_1 \\ \vdots \\ TFSLF_n \end{pmatrix} = TF_i \circ SLF^T \quad (13)$$

Где ' ∘' - операция поэлементного умножения матриц.

F. Оценка качества классификации

Пусть множество документов разбито по категориям. Обозначим v – множество документов, принадлежащих классу, u – множество документов, приписанных классу алгоритмом классификации.

Полнота r (от англ. recall) классификации документов по классу вычисляется как отношение количества документов, принадлежащих к классу, к общему количеству документов, относящихся к данному классу [8, 9]:

$$r(u) = \frac{|u \cap v|}{|v|} \quad (14),$$

Точность p (от англ. precision) классификации документов по классу вычисляется как отношение количества документов, правильно приписанных к классу, к общему количеству документов, приписанных к данному классу [8, 9]:

$$p(u) = \frac{|u \cap v|}{|u|} \quad (15),$$

F-мера (F-measure) объединяет оценки точности и полноты в одну [8, 9]:

$$F(u) = \frac{2 * p(u) * r(u)}{p(u) + r(u)} \quad (16),$$

Если $p(u)=0$ или $r(u)=0$, то тогда и $F(u) = 0$

Для получения сводных характеристик оценок качества классификации по всем классам вводится макро-усреднение характеристик по всем рубрикам [8, 9].

$$Macro - p = \frac{1}{|C|} \sum_{i=1}^{|C|} p(u_i) \quad (17)$$

$$Macro - r = \frac{1}{|C|} \sum_{i=1}^{|C|} r(u_i) \quad (18)$$

$$Macro - F = \frac{1}{|C|} \sum_{i=1}^{|C|} F(u_i) \quad (19)$$

III. Исследование и построение задачи

A. Описание инфраструктуры приложений и данных

Существующая база знаний предприятия представлена следующим набором компонент:

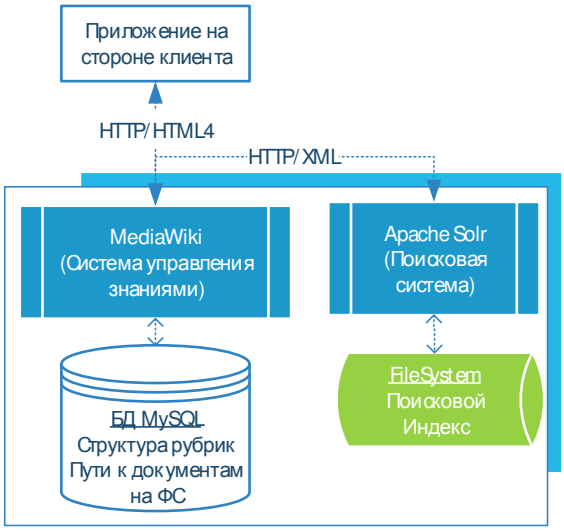

Рисунок 1:Инфраструктура приложений и данных

Уровень представления
Представлен клиентским приложением. Работает через браузер. Написан на HTML, Javascript и CSS.

Уровень бизнес логики
Представлен доработанным под требования предприятия веб-приложением MediaWiki. Основный язык приложения и его расширений – PHP.

Уровень хранения данных
Представлен базой данных MySQL и файловой системой сервера. База данных MySQL хранит операционные данные веб-приложения. Файловая система сервера хранит исходные документы и их распознанное текстовое содержимое.

Внешние веб-приложения
Платформа полнотекстового поиска Apache Solr – основная задача платформы, построение полнотекстового индекса для документов базы знаний.
Приложение расположено на одном виртуальном сервере с веб-приложением MediaWiki.
Сервер распознавания ABBYY Recognition Server – основная задача сервера, принять новые документы, распознать их содержимое, представить документы в виде текста и передать распознанные тексты обратно приложению MediaWiki.

B. Основные сценарии работы с базой знаний

Для создания автоматической системы классификации текстов необходимо знать основные сценарии работы с текстами в рамках сложившейся инфраструктуры.
Необходимые сценарии:

- Загрузка нового документа
- Загрузка нового документа в поисковой индекс

Сценарий 1: Загрузка нового документа в базу знаний MediaWiki
Входные данные document ← новый документ document_id ← идентификатор нового документа category_id ← идентификатор категории document_text ← {} document_path ← полное имя документа на файловой системе
Результат работы Запись в базе данных Распознанный текст на файловой системе Поисковой индекс в поисковой системе
Сценарий 1: **отправить** document на сервер *AbbyyRecognitionServer* 2: document_text ← распознанный текст от *AbbyyRecognitionServer* 3: **сохранить** document_id, category_id, document_path в базе данных 4: **сохранить** document_text на файловой системе в файл document_path 5: **отправить** document_text, document_id на поисковой сервер *Apache Solr* a. Поисковой сервер *Apache Solr* добавляет документ в поисковой индекс

- Из приведенного выше сценария следует, что входе загрузки документа задействованы три системы: MediaWiki, AbbyyRecognitionServer и Apache Solr. Важным данными для построения системы классификации являются:
- Идентификатор документа – при обучении системы, позволит идентифицировать документ. При классификации позволит найти документ в базе знаний и присвоить ему установленную категорию.
- Категория документа – нужна при обучении системы.
- Текст документа – нужен для индексации и извлечения термов.

У поисковой системы и системы классификации текстов есть общий этап – индексация данных. Поисковая система Apache Solr для построения поискового индекса использует текст документа на этапе индексации данных. Таким образом, можно воспользоваться уже существующим поисковым индексом для построения классификатора.
У данного подхода есть ряд преимуществ:

- Повторное использование поискового индекса.
- Нет необходимости разрабатывать свой этап индексации с нуля.
- Экономия вычислительных ресурсов.
- Качественное разбиение термов.

Ниже приведен сценарий загрузки документа в поисковой индекс.

Сценарий 2: Загрузка нового документа в поисковой индекс
Входные данные document_id ← идентификатор нового документа document_text ← распознанный текст документа document_terms ← {} //вектор слов документа
Результат работы Поисковой индекс в поисковой системе
Сценарий 1: document_terms ← извлечение слов и их частот из document_text 2: document_weights ← взвешивание document_terms по внутреннему алгоритму 3: **добавить** document_id, document_terms, document_weights в поисковой индекс

Поисковой индекс документа представляет собой вектор из пар <ключ>:<значение>, где:

- ключ – значение терма
- значение – частота встречаемости данного терма в тексте

Таким образом, логическое представление документа в поисковом индексе удовлетворяет требованиям системы классификации.

Существует ряд задач, которые необходимо дополнительно решить при использовании поискового индекса:

- Взвешивание термов – веса термов поискового индекса не подходят для получения качественного классификатора.
- Удаление лишних термов – поисковый индекс содержит стоп-слова, последовательности знаков препинания и числовые последовательности, которые ухудшают качество классификации.

IV. Построение модели классификатора

Задача выбора модели классификатора документов сильно зависит от предметной области и характеристик рассматриваемых текстов. Реализации существующих методов машинного обучения являются универсальными и не учитывают специфики предметной области. Поэтому для выбора подходящего классификатора выбран экспериментальный подход.

Классификаторы будут оцениваться по общепринятым методам оценки полноты, точности и f-меры, которые были описаны в пункте «Оценка качества классификации».

Для построения классификатора необходимо реализовать следующие компоненты:

- Индексатор документов – отвечает за предобработку данных.
- Классификатор документов – обучение и классификация документов.

Так как существуют статистические методы взвешивания термов, которые дают хороший результат при классификации для малых корпусов документов, в качестве логического представления документа на этапе индексации решено использовать поисковой индекс.

A. Классификатор

Классификатор документа будет выбран с учетом результатов теста методов классификации на экспериментальных данных.

Для классификации будут использованы алгоритмы библиотеки SciKit-Learn, так как:

• Библиотека предоставляет реализации всех необходимых алгоритмов классификации.

• Библиотека предоставляет средства для визуализации и анализа полученных данных.

В эксперименте использованы следующие классификаторы:

1. Классификация методом Байеса:
   - MultinominalNB – классическая реализация метода Байса для многоклассовой классификации.
   - BernoulliNB – в отличие от метода многоклассовой классификации Байеса не учитывает частоту вхождения терма, учитывается только факт вхождения в вектор – есть или нет.
2. Классификация методом опорных векторов:
   - LinearSVC – реализация метода опорных векторов с линейным ядром.
3. Классификатор Роше:
   - NearestCentroid – реализация метода Роше без дополнительных параметров.
4. Метод ближайших соседей
   - KneighborsClassifier – с числом соседей равным 10
5. Случайный лес – RandomForestClassifier с 75 деревьями решений.

B. Полученные результаты

Ниже приведены результаты сравнения комбинаций методов взвешивания и классификации документов.

Таблица 1: Результаты сравнения комбинаций методов взвешивания и классификации документов

Классификатор	F-macro			Лучший метод
	TF-IDF	TF-SLF	Лучшая оценка	
LinearSVC	0,681	0,869	0,869	TF-SLF
RandomForest	0,721	0,868	0,868	TF-SLF
MultinominalNB	0,741	0,865	0,865	TF-SLF
BernoulliNB	0,655	0,742	0,742	TF-SLF
NearestCentroid	0,714	0,515	0,714	TF-IDF
Knn	0,443	0,416	0,443	TF-IDF
Лучшая комбинация методов по оценке F-macro				
LinearSVC	0,681	0,869	0,869	TF-SLF

Таким образом, из полученных результатов следует, что использование метода взвешивания TF-IDF для классификации документов рассматриваемой базы знаний дало плохой результат. Самая высокая оценка F-macro для метода TF-IDF получена с использованием классификатора Роше (NearestCentroid).

Метод взвешивания TF-SLF позволил увеличить оценку F-macro для классификаторов: SVM (метод опорных векторов), RandomForest (случайный лес), Naïve Bayes (Байесовский классификатор)

Ниже представлены результаты оценки классификаторов в разрезах полноты и точности.

Таблица 2: Оценки классификаторов в разрезах полноты и точности

Классификатор	P-Macro (Точность)	R-Macro (Полнота)
LinearSVC	0.88	0.87
RandomForest	0.87	0.87
MultinominalNB	0.92	0.86
BernoulliNB	0.89	0.70
NearestCentroid	0.50	0.43
Knn	0.76	0.49

Из результатов оценки полноты и точности алгоритмов следует, что самая большая точность - у метода классификации MultinominalNB, самый высокий результат полноты - у LinearSVC и RandomForest классификаторов.

В качестве метода классификации для базы знаний выбран метод MultinominalNB, так как он показал лучшую оценку точности 0.92 и сравнительно хорошую полноту 0.86.

Детальные оценки по рубрикам для MultinominalNB:

Таблица 3: Детальные оценки по рубрикам для MultinominalNB

Идентификатор	Точность	Полнота	F1-scor	Документов в

			e	тесте
136	1.00	0.71	0.83	17
140	1.00	0.68	0.81	25
149	0.62	1.00	0.77	5
152	1.00	1.00	1.00	2
158	0.56	1.00	0.71	10
162	1.00	1.00	1.00	14
164	0.71	1.00	0.83	5
178	1.00	1.00	1.00	15

Список значимых термов при классификации методом MultinominalNB:

Таблица 4: Список первых значимых термов при классификации методом MultinominalNB

Id	Название категории	Топ термов
136	Теория и моделирование компьютерных сетей	Маршрутизация, пакет, деть, коммутатор, спецификация, модель, пкс, политик, сет
140	Управление сетевой инфраструктурой	Mininet, and, управление, network, сет, the, сеть, цод, сетевой, quot
149	Обеспечение безопасности ПКС сетей	Сет, сеть, решение, solutions, платежей, mobile, платёж, мобильный, payment, платеж
152	Облачные технологии	Hardware, small, aws, techcrunch, techcrunchcom, opencloud, required, offers
158	Тестирование ПКС	Segment, средство, цод, сегмент, the, пкс, network, сет, сеть, сетевой
162	Семинары, школы, курсы	Admin, file, openstack, compute, server, the, keystone, sudo, nova, swift
164	Материалы конференций	Tom, update, portal, nfv, marshall, brinn, gec, слайд, geni, презентация
178	Другое	Цпикс, and, computer, specialists, the, onlab, lab, communications, журнал, acm

Из таблицы 4 не сложно заметить, что для категорий 136, 149, 158 самые значимые термы пересекаются: сет, сеть, сетевой. Этим пересечением можно объяснить снижение точности классификации для данных категорий.

C. Итоговая модель

Итоговая модель классификации данных использует метод взвешивание TF-SLF и метод классификации MultinominalNB. Основным преимуществом данной модели является её высокая точность для рассматриваемой базы знаний.

С ростом базы и появлением новых терминов и документов классификация может ухудшиться. Поэтому по мере роста базы знаний следует переобучать классификатор. На данный момент в базе знаний очень мало документов.

V. Описание основных компонентов системы

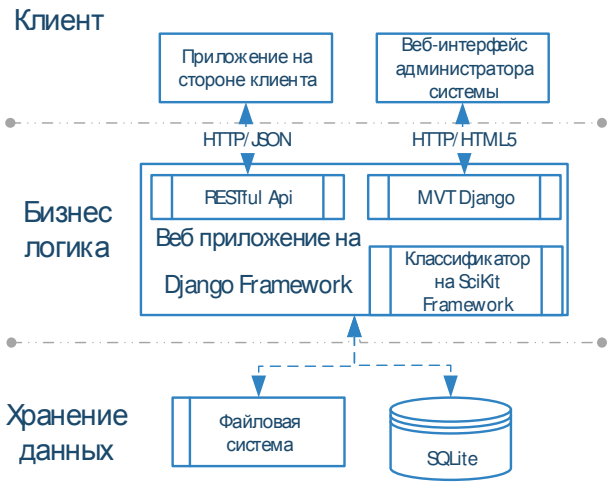

Рисунок 2: Основные компоненты системы

Уровень представления

- Клиентское приложение для администрирования системы.
  - Написано на HTML, JavaScript и CSS.
- RestFull Api для загрузки, изменения и получения данных о документах.

Уровень бизнес-логики

Уровень бизнес логики обрабатывает поступающую от пользователей информацию.

Данный уровень построен в соответствии со стилем построения архитектуры REST.

REST (Representational State Transfer) – стиль построения архитектуры распределенного приложения. REST используют для построения приложений, в которых клиенты могут отправлять запросы службам, т.е. реализация подхода «клиент-сервер». Данные в REST должны передаваться по протоколу HTTP в виде небольшого количества данных в одном из форматов: HTML, Xml, JSON. [10]

Веб-приложение построено с использованием языка Python и веб-фреймворка Django.

Django – свободный фреймворк для веб-приложений. Использует шаблон проектирования MVC (model-view-controller)

Для реализации RESTful служб используется подключаемый модуль rest-framework.

Использование фреймворков Django и rest-framework оправдано их свойствами. Django, в отличии от множества существующих веб-фреймворков, предоставляет удобный механизм явной конфигурации обработчиков URL при помощи регулярных выражений, этот механизм не зависит от структуры контроллеров приложения. Таким образом естественным образом возможно реализовать архитектуру RESTful приложения. Структурой URL управляет основное веб-приложение на Django, при обращении к URL ответственных за веб-службы управление передается контролерам служб, реализованных с использованием rest-framework.

Уровень хранения данных

Уровень хранения данных представлен базой данных SQLite.

SQLite – компактная встраиваемая реляционная база данных. SQLite предоставляет библиотеку, с которой программа компонуется и движок SQLite становится частью программы. В результате для обмена используются вызовы api функций библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу.

Несколько процессов или потоков могут одновременно читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается.

VI. Заключение

В рамках работы были исследованы и опробованы методы машинного обучения для решения задачи классификации текстов небольшой базы знаний. Получены следующие теоретические результаты для базы с числом документов меньше пятисот и сильно варьирующимся числом документов внутри каждой категории:

- Применение метода TF-SLF для извлечения термов позволяет улучшить качество классификации на многоязычных документах, в рамках корпуса с числом документов меньше 500.
- Применение многоклассовой классификации Байеса совместно с применением метода TF-SLF позволяет получить лучший результат по точности классификации текстов.

Предложено представление метода TF-SLF в матричном виде и реализован его алгоритм на numpy, расширении языка python для быстрых вычислений над матрицами.

По результатам исследований базы знаний и инфраструктуры предложена и реализована система автоматической классификации документов в виде веб-приложения с сервисом RESTful api. В рамках разработанной системы предложен сценарий использования существующего поискового индекса базы знаний для классификации данных.

Библиография

[1] Gantz J., Reinsel D., The Digital Universe: As the Economy Contacts, the Digital Universe Expands, 2009. - 6 с.

[2] Находимость корпоративных данных: обзор опыта пользователей ECM-систем // Abbyy, Docflow, 2014. - 5 с.

[3] Корпоративная база знаний, 2010 http://enterprisekb.ucoz.ru/news/korporativnoj_bazy_znanij/2010-11-25-1.

[4] Лифшиц Ю., Алгоритмы для интернета: Автоматическая классификация текстов, 2006. – 2 с.

[5] Агеев М. С., Методы автоматической рубрикации текстов, основанные на машинном обучении знаниях экспертов, 2004. – С. 6.

[6] Губин М. В. Модели и методы представления текстового документа в системах информационного поиска, 2005. – С. 11-12.

[7] Rehman A., Haroon A., Saeed M., Feature Extraction for Classification of Text Documents, 2012. – С. 233-235.

[8] Токарева Е. И. Иерархическая классификация текстов, 2010. – С. 11-12.

[9] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schutze, «An Introduction to Information Retrival» // Cambridge UP, 2009. – С. 152-156.

[10] Flanders J., Введение в службы RESTful с использованием WCF, 2009, http://msdn.microsoft.com/ru-ru/magazine/dd315413.aspx.

Text Analytics for Enterprise Knowledge Base

Popkov Maxim Ivanovich

Abstract— **The paper discusses machine learning methods for data classification. Author examined indexing techniques, weighing and classification of documents in the enterprise knowledge base. The paper describes the metrics for comparing classifiers and presents the results for the existing infrastructure.**

Author examined the way to use the search index for document classification, developed a web application and the algorithms for text classification.

Key words—**Text classification, search index, web service.**