ВЫСОКОТОЧНЫЙ МЕТОД РАСПОЗНАВАНИЯ КОНЦОВ ПРЕДЛОЖЕНИЙ

А. С. Кудинов (al.kudinov@corp.mail.ru)

А. А. Воропаев (voropaev@corp.mail.ru)

А. Л. Калинин (kalinin@corp.mail.ru)

Проект Поиск@Mail.Ru, Москва, Россия

В статье описывается метод применения машинного обучения в задаче распознавания концов предложений. Предлагаемый способ успешно решает проблему идентификации знаков препинания, таких как точка и др., которые не являются знаками конца предложения. Несмотря на сравнительно небольшой объем обучающей выборки, подготовленной вручную, способ демонстрирует точность не менее 99% на среднестатистическом web-документе.

Ключевые слова: машинное обучение, конец предложения, знаки препинания, идентификация знаков препинания.

A HIGH PRECISION METHOD FOR THE RECOGNITION OF SENTENCE BOUNDARIES

A. S. Kudinov (al.kudinov@corp.mail.ru)

A. A. Voropaev (voropaev@corp.mail.ru)

A. L. Kalinin (kalinin@corp.mail.ru)

Project Search@Mail.Ru, Moscow, Russian Federation

We present a machine-learning method of sentence boundary recognition. The approach successfully identifies punctuation marks, such as periods or question marks that are not sentence boundary markers. In spite of a relatively small initial learning set (which was prepared manually), the accuracy of this approach appears to be no less than 99 % when applied to an average web document. The method is based upon the decision tree technique combined with a tiny set of manually constructed rules that play the role of classifi cation features. The rules are built using a dedicated declarative language, which is briefl y described. A comparison of accuracy of the approach with two freely accessible software products is provided. According to our

estimates, the algorithm provides good enough performance to be used in real-time environment such as indexer component of a web search engine. It can also be used to produce large learning sets to train faster machine learning models such as the maximum entropy model.

Key words: machine learning, sentence boundary, punctuation marks, punctuation marks identification.

Введение

В процессе разработки систем автоматической обработки естественных текстов часто возникает задача о корректном разбиении текста на предложения. В отдельных случаях её решение имеет принципиальное значение:

1. При генерации контекстов словоупотребления, например, для задач снятия морфологической омонимии [3] или семантической омонимии [4].
2. В задачах автоматической классификации текстов (рубрицирования) [5], где структурными единицами документа являются предложения, заглавия, названия колонок в таблицах, элементы списка и др.
3. При реализации корректного графематического и синтаксического машинного анализа текста [6].
4. В частности, при построении сниппетов поисковых системах: найденные слова, употребленные в пределах одного предложения или его сложноподчиненной части могут иметь более высокий ранг по сравнению с теми же словами, употребленными в соседних предложениях.

и др.

Приведенные выше примеры требуют уточнения касательно того, что конкретно мы называем предложением. Речь идёт не о традиционном определении предложения, подразумевающим грамматически организованную единицу речи, обладающую смысловой и интонационной законченностью. Предложениями мы считаем более широкий класс единиц речи, включающий не столько «законченные» элементы, сколько сочетания слов, наиболее часто встречающиеся в реальных коллекциях документов, которые, в зависимости от задачи, удобно рассматривать как единое целое. Ясно, что подобные фрагменты далеко не всегда обладают смысловой полнотой, впрочем, иногда они могут обладать и некоторой избыточностью. Например, отдельный элемент библиографического списка удобнее рассматривать как целую единицу текста, не смотря на наличие в нём формально корректных концов предложения. Другим примером может быть заголовок таблицы, который также является целой единицей текста, хотя и заканчивается двоеточием.

На практике исходная задача разделяется на две подзадачи. Первая подзадача заключается в том, чтобы выяснить, является ли знак концом предложения. Вторая подзадача (см., например, [2]), гораздо более сложная как в реализации, так и вычислительно, подразумевает определение мест в тексте, где делитель

предложения был пропущен, например, по ошибке. К счастью, в решении второй подзадачи, как правило, нет острой необходимости, поскольку «подразумеваемые» концы предложений встречаются существенно реже, чем явные.

Как следует из данного выше определения, концы предложений могут обозначаться точками, знаками вопроса и восклицания, символами конца абзаца, многоточиями и, в отдельных случаях, двоеточиями. Основные проблемы создаёт символ точки, поскольку, он также используется в сокращениях и литеральных обозначениях — датах, шифрах, адресах электронной почты, web-адресах и т. п.

Для определения, является ли точка в данном контексте концом предложения, применяются различные методы, сложность которых в целом зависит от типа входных данных и требований к качеству результата. В силу своей простоты, наиболее распространены способы, предполагающие точное постулирование понятий начала и конца предложения (см., например, [6]), а также применение таблиц стандартных сокращений (в том числе генерируемых автоматически — см. [1]), или регулярных выражений — практика, получившая широкое распространение в среде разработчиков ПО. Подобные способы зачастую показывают хорошие результаты, но в достаточно специфичных случаях, на которые они рассчитаны. К примеру, для разметки газетных корпусов весьма эффективным оказывается метод автоматического распознавания аббревиатур [1]. Отметим, что для выполнения машинного обучения авторам метода потребовалось проанализировать весьма объемные текстовые корпусы (до 1 млн. слов для английского языка).

Недостатком непосредственного использования таблиц сокращений в качестве достаточного признака является неприменимость к случаям нестандартных (авторских) сокращений, кроме того, стандартные сокращения могут находиться в конце предложения, а некоторые сокращения, например, «и т. п.», «и др.» часто обозначают конец предложения явно (что примечательно, данное предложение является исключением).

Очевидно, что информация о том, является ли знак концом предложения, хранится в его контексте. Также очевидно, что эта информация выводится из контекста по неким определенным правилам. Сформулировать эти правила, основываясь только на собственном опыте, оказывается не так уж просто, особенно с учётом того, что их требуется представить в форме алгоритма. Тем не менее, оказывается, что можно автоматически вывести правила, позволяющие различать знаки концов предложений более чем с 99% точностью.

Метод

Предлагаемый нами метод заключается в первоначальном создании небольшого набора базовых правил (порядка 40 штук) и последующем автоматическом построении классификатора, опирающегося на результаты применения этих правил. Базовые правила делятся на два типа — подстановки и комбинации. Подстановки, в общем случае, задаются регулярными выражениями и проверяют конкретные простые признаки контекста такие как, например, «пробелы справа» или «одна прописная буква слева» и т. п. Комбинации являются

алгебраическими конструкциями, строящимися из подстановок, например, «прописная буква слева» + «титул справа» (титул — слово, начинающееся с прописной буквы). Каждое из базовых правил, будучи примененным к конкретному контексту, возвращает число начисленных очков, обычно это 0, -1 или +1. Конечный результат вычисляется из вектора очков посредством специального классификатора, обученного на заранее размеченном наборе текстов.

Обучение классификатора

Для построения классификатора нами был применен алгоритм машинного обучения на основе деревьев принятия решений. Начальным этапом обучения классификатора стала подготовка первичной выборки — сравнительно небольшого текста, порядка 1000 предложений, содержащего значительное количество «трудных» случаев: стандартных и авторских сокращений, как в середине, так и в конце предложения, специальных обозначений, дат, web-адресов, адресов электронной почты и т. п.

Выборка была подготовлена таким образом, чтобы простейший алгоритм (baseline), считающий делителями все точки, вопросительные и восклицательные знаки, ошибался по возможности в максимальном количестве случаев. Всего в подготовленном подготовленном тексте было 3220 потенциальных делителей, т. е. знаки являющиеся действительными концами предложений (согласно нашему определению) составляли около 30 %. Таким образом, на первичной выборке baseline ошибался в 70 % случаев.

Обученный на первичной выборке классификатор затем последовательно применялся к дополнительно отобранным документам различных типов — web-страницам, художественным и формальным текстам. Случаи, когда классификатор ошибался или «сомневался» поступали частично в обучающий и частично в проверочный наборы следующей итерации, после чего производилось переобучение. Целью этих итераций была оценка предельно возможной точности метода, которая по результатам тестирования составила 98,7 ± 0,4 %. Окончательная проверка производилась вручную на 10 случайных web-документах, содержащих в сумме 2681 эпизод, из которых 28 оказались ошибочно не разделенными и 7 ошибочно разделенными. Всего потребовалось 6 итераций переобучения, за которые была накоплена финальная обучающая выборка порядка 10 тыс. эпизодов с отношением количества знаков-делителей предложений к знакам, не являющимся делителями, как 2 к 3. Попытки продолжать обучение точность не повышали, что связано, как выяснилось, с существованием случаев, когда знак препинания должен быть классифицируем как конец или не конец предложения, исходя из семантических признаков, т. е. требующих смысловой идентификации элементов предложения (см. пример 1).

Пример 1. *Случай, когда точка является концом предложения по семантическим признакам.*
«Описание модели см. в А21. К-2301 т. IV, стр. 45. С 2003 г. изменена номенклатура.»

Принятие решения в примере 1 затруднено, поскольку в контексте каждой точки находятся последовательности, которые с точки зрения набора правил могут являться как стандартным или авторским сокращением, так и частью шифра. В подобных случаях разбиение предложения нашим классификатором, как правило, не происходит.

Сравнение с существующими методами

Для получения сравнительных оценок точности и производительности нашего метода мы воспользовались двумя открытыми разработками, предлагающими свои решения задачи определения концов предложений — модулем графематического анализа из проекта «АОТ» [6], основанном на эвристике, и компонентом Sentence Boundary Detector из проекта OpenNLP, использующим принцип максимальной энтропии [7]. Последний компонент мы обучили на той же финальной выборке, по которой обучался наш классификатор. В качестве тестовой коллекции была использована коллекция документов, полученных с новостных и аналитических сайтов, таких как news.mail.ru, lenta.ru, rian.ru, rbc.ru, wciom.ru и др., содержащая большое количество инициалов, сокращений, дат и других обозначений, включающих символы пунктуации. Для получения текстового содержания статей выкачивались в основном препринты (версии документов с минимальной разметкой, подходящей для печати). Очистка документов от включений тэгов и HTML-сущностей производилась посредством html-парсера, используемого в индексаторе нашей поисковой системы. Поскольку графематический модуль проекта «АОТ» все переносы строк однозначно считает концами предложений (абзацами), все переносы строк из документов тестовой коллекции были заменены пробелами. Результаты приведены в таб. 1.

Таблица 1. Сравнительные характеристики методов выделения концов предложений

	OpenNLP / Sentence Boundary Detector.	**Графематический модуль проекта «АОТ».**	**Наш классификатор.**
Алгоритм:	принцип максимальной энтропии	эвристический	набор правил и дерево принятия решений
Число знаков в обучающей выборке:	9820	Обучение не производилось.	9820 (та же выборка)
Число потенциальных делителей в тестовой выборке:	Всего: 3087 вхождений: « . » — 2980 вх., « ? » — 37 вх., « ! » — 70 вх.		

[1] Указанное время не должно расцениваться как показатель производительности графематического анализатора, входящего в состав АОТ, поскольку использованный нами компонент помимо графематического анализа также производил синтаксическую разметку входного текста.

	OpenNLP / Sentence Boundary Detector.		Графематический модуль проекта «АОТ».		Наш классификатор.	
Общее число знаков, признанных делителями:	2760		2522		2069	
Размер случайной выборки для ручной проверки:	500 элементов					
Время выполнения:	0,42 с		12,1 с[1]		0,72 с	
	Верные:	**Неверные:**	**Верные:**	**Неверные:**	**Верные:**	**Неверные:**
Число разбиений:	339	161	386	114	499	1
Число слияний:	455	45	462	38	497	3
Общий процент ошибок (*Error rate*):	41,2%		30,4%		0,8%	
Точность (*Precision*):	0,678		0,772		0,998	
Полнота (*Recall*):	0,883		0,910		0,994	
F-мера Ван Ризбергена:	0,767		0,835		0,996	

Отметим, что невысокая точность метода, использующего принцип максимальной энтропии, в целом обусловлена небольшим объемом обучающей выборки и может быть значительно улучшена.

О способе описания правил классификатора

Подбор правил производился вручную, для чего использовался специально разработанный декларативный язык. Каждое декларируемое правило снабжается уникальным именем, посредством которого на него могут ссылаться другие правила. При объявлении указывается класс правила, задающий способ вычисления, и, собственно, вычисляемая формула. Классом правила может являться регулярное выражение, алгебраическая конструкция и сам классификатор — объект, опрашивающий ассессора или получающий информацию о графематической разметке из файла.

Пример 2. Объявление правила, распознающего сокращения «т. к.», «т. е.», «т. н.».

```
ABBR3a_L := RegEx(L) <- $(open)т\z
ABBR3a_R := RegEx(R) <- \A$(s)*[енк]\.
ABBR3b_L := RegEx(L) <- $(open)т\.$(s)*[енк]\z
ABBR3 := Rule() <- (ABBR3a_L & ABBR3a_R ) | ABBR3b
```

В представленном выше примере объявлены четыре правила, три из которых являются регулярными выражениями, причем два применяются к левому контексту (класс RegEx(L)) и одно к правому (класс RegEx(R)). Последнее

правило выражает логическую связь между остальными правилами посредством операторов «&» (логическое И) и «|» (логическое ИЛИ).

Производительность и оптимизация

Исследование производительности алгоритма показало, что основная нагрузка приходится на обслуживание регулярных выражений, что не удивительно, поскольку большинство правил относятся к этому типу. В общем случае, если применять регулярные выражения ко всему левому и правому контексту, то сложность алгоритма получается квадратичной. В особенности это касается левого контекста, поскольку в этом случае затруднена привязка регулярного выражения к концу проверяемого диапазона посредством соответствующего нетерминала (\z).

Исследование возможности задать максимально допустимый диапазон для проверки показало (см. Рис. 1), что для русскоязычных текстов достаточно 28 символов слева и 16 справа (при этих ограничениях на нашем наборе правил не происходила потеря точности).

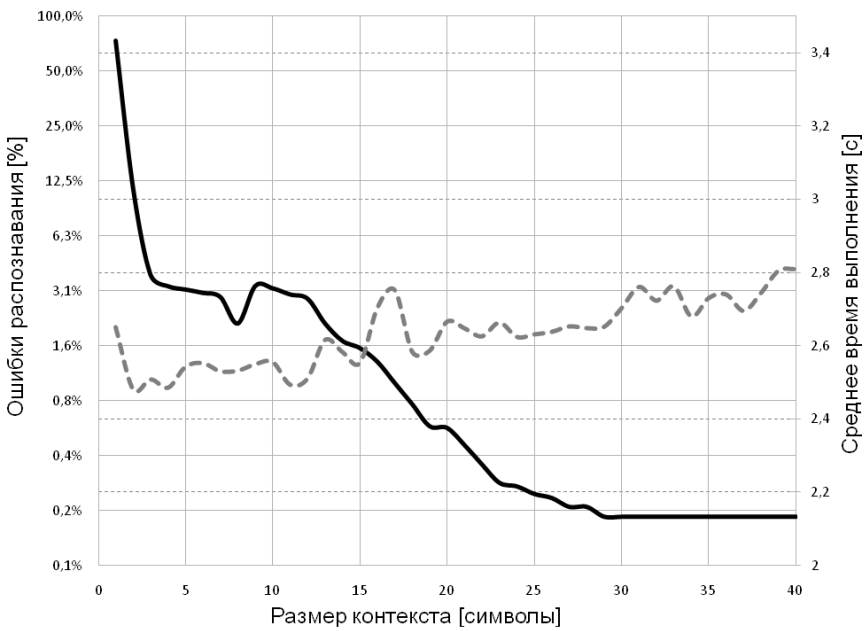

Рис. 1. Зависимость точности и скорости от размера контекста (для 50 тыс. эпизодов)

Для повышения производительности также были применены дополнительные приёмы:

1. По набору правил автоматически сгенерирован эквивалентный программный код, снимающий вычислительную избыточность, связанную с обращениями к правилам по имени, а также виртуальными вычислениями. Эффект +30 % производительности.
2. Вычисление не всех правил, а лишь тех, которые необходимы классификатору (по обратному запросу). Эффект +25 % производительности.
3. Кэширование значений вычисленных правил на период разбора одного эпизода. Эффект +5 % производительности.

В сумме применение всех перечисленных приёмов позволило получить приемлемую скорость для процесса индексирования в нашей поисковой системе.

Наиболее значимые правила

Значимость отдельных правил определяется автоматически в процессе обучения. Грубо говоря, чем выше степень корреляции оценки, получаемой некоторым правилом, с требуемым ответом, тем выше это правило будет в списке опроса, и тем чаще его выполнение будет проверяться. По итогам проведенных экспериментов, к наиболее значимым правилам относятся

1. «тип разделителя», 3 правила — определяют соответственно точку, знак вопроса и знак восклицания;
2. «пробелы справа/слева» — определяют наличие пробельных символов справа и слева от разделителя;
3. «символ пунктуации справа/слева»;
4. «цифра справа/слева»;
5. «прописная/строчная буква справа/слева»;
6. «открывающая/закрывающая скобка справа/слева»
7. «стандартные сокращения»;
8. «неизвестные сокращения общего вида ххх.-хх. хх.» и т. д.

Об алгоритме построения классификатора

В качестве основы классификатора мы использовали деревья принятия решений, широко применяемые в машинном обучении [8, 9, 10]. Пример реального дерева принятия решения из нашего классификатора показан на Рис. 2 (мы приводим фрагмент, так как всё дерево слишком велико). В узлах дерева находятся условия проверки выполнения правил. Запись вида «20 < 0.5» означает, что если правило №20 набрало менее 0.5 очков, то нужно перемещаться в левое поддерево, а в противном случае в правое поддерево. В листьях дерева хранятся оценки вероятности того, что данный знак препинания является концом предложения (чем больше значение, тем выше вероятность).

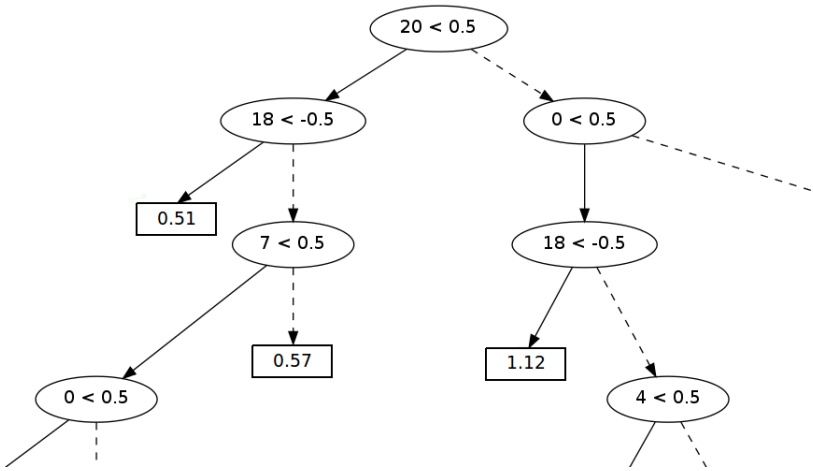

Рис. 2. Фрагмент дерева принятия решений

В приведенном примере (см. Рис. 2) классификатор принимает решение, обращаясь к правилам, перечисляемым ниже:

1. № 0 — разделитель является точкой;
2. № 4 — пробелы справа;
3. № 7 — прописная буква слева;
4. № 18 — многоточие (+2 для первой точки, +1 для средних, -1 для последней);
5. № 20 — титул справа (титул — слово, записанное с прописной буквы).

К сожалению, одно дерево не может обеспечить приемлемое качество классификации. Для того, чтобы решить эту проблему, в качестве классификатора мы используем комбинацию нескольких десятков относительно простых деревьев. Каждое дерево строится так, чтобы максимально скомпенсировать ошибку работы остальных деревьев классификатора (такой подход известен в специальной литературе под названием boosting — см. [11, 12]). Дополнительно, для улучшения качества работы классификатора в целом мы использовали технику bootstrapping, которая заключается в том, что для построения каждого дерева используется случайное подмножество обучающего множества, меняющееся от дерева к дереву. В нашем случае было автоматически построено 540 деревьев, каждое из которых содержит до 8 уровней и от 300 до 350 листьев. Полученный таким образом композитный классификатор показывает значительно более высокую точность, чем отдельно взятое дерево.

Пример разбиения

В таб. 2 приведены оценки, сделанные классификатором для части web-документа. Порог отсечения (равный 0,64) выбирался методом наименьших

квадратов для оценок, полученных по обучающей выборке. Смещение порога объясняется несимметричностью обучающей выборки относительно числа случаев, когда знак является разделителем и когда не является.

Таблица 2. Пример вывода размечающей программы для «сложного» web-документа

Левый контекст:		Правый контекст:	Оценка (0 — не конец предложения, 1 — конец).	
/римером в круглых скобках	.	##Нумерация рисунков такж/	Да	1.000000
/я рисунков также сквозная	.	Подписи под рисунками до/	Да	1.000000
/лжны начинаться так: «Рис	.	<Номер рисунка> <Названи/	**Нет**	**1.004000 (ошибка)**
/унка>» и набраны курсивом	.	##Подзаголовки должны быт/	Да	1.000000
/браны полужирным курсивом	.	При делении на разделы р/	Да	1.000000
/разделов и подразделов (1	.	1., 1.2., 1.3, 2.1 и т.д./	Нет	0.001357
/зделов и подразделов (1.1	.	, 1.2., 1.3, 2.1 и т.д.)./	Нет	0.020330
/ов и подразделов (1.1., 1	.	2., 1.3, 2.1 и т.д.).##Сн/	Нет	-0.000760
/ и подразделов (1.1., 1.2	.	, 1.3, 2.1 и т.д.).##Снос/	Нет	0.006150
/одразделов (1.1., 1.2., 1	.	3, 2.1 и т.д.).##Сноски т/	Нет	-0.000760
/делов (1.1., 1.2., 1.3, 2	.	1 и т.д.).##Сноски также /	Нет	-0.000760
/(1.1., 1.2., 1.3, 2.1 и т	.	д.).##Сноски также должны/	Нет	0.051560
/.1., 1.2., 1.3, 2.1 и т.д	.	).##Сноски также должны б/	Нет	-0.006359
/., 1.2., 1.3, 2.1 и т.д.)	.	##Сноски также должны быт/	Да	1.000000
/и помещены внизу страницы	.	##Список литературы оформ/	Да	1.000000
/через запятую, год, точка	.	Затем указываются том (Т/	Да	1.000000
/ Затем указываются том (Т	.	), номер издания (№), стр/	Нет	-0.011740
/ издания (№), страницы (С	.	).##Пример оформления:##1/	Нет	-0.011740
/здания (№), страницы (С.)	.	##Пример оформления:##1. /	Да	1.000000
/).##Пример оформления:##1	.	Крейдлин Г. Е. Невербальн/	**Нет**	**0.498900**
/формления:##1. Крейдлин Г	.	Е. Невербальная семиотика/	Нет	-0.003087
/рмления:##1. Крейдлин Г.Е	.	Невербальная семиотика //	Да	1.000000
/вербальная семиотика // М	.	: Новое литературное обоз/	Нет	0.291300
/ературное обозрение, 2002	.	#2. Якобсон Р. О. О лингви/	Да	0.661100
/турное обозрение, 2002.#2	.	Якобсон Р. О. О лингвисти/	**Нет**	**0.498900**
/рение, 2002.#2. Якобсон Р	.	О. О лингвистических аспе/	Нет	0.001101
/ние, 2002.#2. Якобсон Р.О	.	О лингвистических аспект/	Да	1.000000
/ в зарубежной лингвистике	.	М.: 1978. С.16–24.#Остал/	Да	1.000000
/зарубежной лингвистике. М	.	: 1978. С.16–24.#Остальны/	Нет	-0.008412
/ной лингвистике. М.: 1978	.	С.16–24.#Остальные парам/	**Нет**	**0.481700**
/ лингвистике. М.: 1978. С	.	16–24.#Остальные параметр/	Нет	0.002148
/истике. М.: 1978. С.16–24	.	#Остальные параметры текс/	**Да**	**0.498900**
/, указанными верстальщику	.	#/	Да	1.000000

Заключение

Примененный нами подход показывает, что совмещение эффективного инструмента для ручного подбора правил с автоматическим классификатором (в частности, на основе деревьев принятия решений) позволяет использовать сравнительно небольшие тренировочные наборы, что, в свою очередь, не только экономит время работы асессоров, но и позволяет сделать их работу менее монотонной. После достижения необходимого уровня точности, метод, в зависимости от реализации, допускает существенную оптимизацию по скорости (в нашем случае до 50%). В случаях, если скорость работы классификатора недостаточно высока для конкретной задачи, с его помощью можно обучать более быстрые модели, требующие, как правило, существенно бо́льших обучающих выборок.

References

1. *Berger A. L., Della Pietra V. J.* 1996. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 22.
2. *Bishop C.* 2006. Pattern Recognition and Machine Learning.
3. *Breiman L., Friedman J. H., Olshen R., Stone C. J.* 1984. Classification and Regression Tree.
4. *Friedman J. H.* 1999. Stochastic Gradient Boosting.
5. *Stevenson M., Gaizauskas R.* 2000. Experiments on Sentence Boundary Detection. Proceedings of the Sixth Conference on Applied Natural Language Processing and First Conference of the North American Chapter of the Association for Computational Linguistics.
6. *Hastie T., Tibshirani R., Friedman J. H.* The Elements of Statistical Learning. Second Edition.
7. *Kiss T., Strunk J.* 2006. Unsupervised Multilingual Sentence Boundary Detection. Computational Linguistics, 32 (4).
8. *Kobritsov B. P., Liashevskaia O. N., Shemanaeva O. Iu.* 2005. Superficial Filters for Semantic Omonymy Settlement in the Text Corpus [Poverkhnostnye Fil'try dla Razresheniia Semanticheskoi Omonimii v Tekstovom Korpuse]. Komp'iuternaia Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoi Konferentsii "Dialog 2005" (Computational Linguistics and Intelligent Technologies: Proceedings of the International Conference "Dialog 2005").
9. *Petrovskii M. I., Glazkova V. V.* 2007. Machine Learning Algorythms for Electronic Documents Analysis and Rubrication Target [Algoritmy Mashinnogo Obucheniia dla Zadachi Analiza I Rubrukatsii Elektronnykh Dokumentov]. Vychislitel'nye Metody i Programmirovanie, 8.
10. *Sokirko A.* Description of the Graphematic Unit of the Project "AOT" [Opisanie Grafematicheskogo Modulia Proekta "AOT"], available at: http://www.aot.ru/docs/graphan.html

11. *Zelenkov Iu. G., Segalovich Iu. A., Titov V. A.* 2005. Probabilistic Model of Morphological Omonymy Elimination on the Base of Normalizing Substitutions and Adjacent Words Positions [Veroiatnostnaia Model' Sniatiia Morfologicheskoi Omonimii na Osnove Normalizuiushchikh Podstanovok I Pozitsii Sosednikh Slov]. Komp'iuternaia Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoi Konferentsii "Dialog 2005" (Computational Linguistics and Intelligent Technologies: Proceedings of the International Conference "Dialog 2005").