ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет Информационных технологий и программирования

Направление (специальность) Прикладная математика и информатика

Квалификация (степень) магистр прикладной математики и информатики

Специализация _____

Кафедра компьютерных технологий Группа 6539

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

НА ТЕМУ

Извлечение отношений из текстов на естественном языке

Автор работы Чернявский И.И. (подпись)
(Фамилия, И., О.)

Руководитель Шалыто А. А. (подпись)
(Фамилия, И., О.)

К о н с у л ь т а н т ы :

а) По экономике и организации производства _____ (подпись)
(Фамилия, И., О.)

б) По безопасности жизнедеятельности и экологии _____ (подпись)
(Фамилия, И., О.)

в) _____ (подпись)
(Фамилия, И., О.)

К з а щ и т е д о п у с т и т ь
Зав. Кафедрой Васильев В.Н. (подпись)
(Фамилия, И., О.)

“___” _____ 2012 г.

Санкт-Петербург, 2012 г.

Аннотация

В настоящей работе исследуется задача извлечения отношений из текстов на естественном языке. Приводится формальная постановка задачи, рассматриваются подходы к ее решению. В работе реализована система извлечения отношений для текстов на русском языке, приводятся результаты экспериментов, проводится исследование эффективности различных алгоритмов извлечения отношений.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задача обработки текстов на естественных языках (*natural language processing* [1]) заключается в автоматическом анализе и синтезе текстов, а также извлечении интересующей информации из них.

В широком смысле данная задача заключается в обучении компьютера пониманию естественного языка, созданию систем, способных к взаимодействию с пользователем посредством диалога на естественном языке.

К подзадачам, которые решаются в данной области, относятся:

- Анализ текста;
- Синтез и распознавание языка;
- Машинный перевод;
- Извлечение информации и др.

К более низкоуровневым подзадачам можно отнести:

- Определение частей речи;
- Определение имен собственных;
- Определение синтаксической структуры предложения и др.

Началом исследований в области обработки текстов принято считать 50-е годы XX века. Одним из ключевых событий является работа Алана Тьюринга, в которой был предложен тест-критерий интеллектуальности систем (позднее известный как тест Тьюринга). В рамках данного теста система считалась интеллектуальной, если человек, общающийся с ней на естественном языке, не мог отличить систему от человека. В дальнейшем возможность общаться с человеком на естественном языке стала

рассматриваться как неотъемлемая и необходимая характеристика интеллектуальной системы.

На заре существования области был создан ряд успешных систем. К ним можно отнести, например, программу ELIZA, созданную в 60-х годах и способную вести диалог с пользователем.

Вплоть до 80-х годов системы обработки текстов в основном были основаны на написанных вручную правилах. Современные исследования в области обработки текстов основываются на применении методов машинного обучения и лингвистики. Применение методов машинного обучения стало возможным во многом благодаря увеличению вычислительной мощности компьютеров, появлению различных моделей машинного обучения, созданию объемных лингвистических корпусов.

В данной работе рассматривается задача из области обработки текстов на естественных языках – извлечение отношений. В рамках данной задачи требуется выделить из предложений *отношения* – осмысленные утверждения о субъектах (действующих лицах) предложения.

Извлеченные отношения можно использовать для построения систем *онтологий* – баз знаний об окружающем мире, готовых к использованию в автоматических системах обработки текстов.

В работе проводится исследование задачи и реализуется система извлечения отношений из текстов на русском языке. Рассматриваются известные подходы к решению задачи. Проводится сравнение различных алгоритмов работы с текстом для задачи извлечения отношений. Рассматриваются несколько моделей машинного обучения – модель максимальной энтропии и графическая модель CRF и проводится сравнение эффективности данных моделей на первом этапе работы системы. Рассматриваются эвристические алгоритмы и алгоритмы

классификации, и проводится сравнение эффективности данных алгоритмов на втором этапе работы системы. В заключении работы делается вывод об эффективности рассмотренных подходов.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИЗВЛЕЧЕНИЯ ОТНОШЕНИЙ ИЗ ТЕКСТОВ

1.1. Применение машинного обучения для работы с текстами

Первые системы обработки естественного языка основывались на прямом задании множества правил работы системы.

Современные методы в области обработки естественного языка используют парадигму машинного обучения [2]. В рамках современного подхода множество правил строится автоматически путем обучения выбранной *модели*. Как правило, обучение производится на большом объеме размеченных текстов, называемом *лингвистическим корпусом*.

В данной работе рассматриваются модель максимальной энтропии и графическая модель CRF. В следующих разделах будут даны определения данных моделей, приведены алгоритмы обучения и вывода на них.

1.2. Модели максимальной энтропии

1.2.1. Принцип максимальной энтропии

Многие задачи в области обработки текстов на естественных языках можно переформулировать как задачи классификации, в которых требуется определить вероятность данного ответа (*класса*) ***a*** при заданном *контексте* ***b***, обозначаемую $p(a|b)$. Так, например, в задаче определения имен собственных в предложении будет всего два класса – имя собственное или нарицательное, а контекстом может быть слово, его часть

речи и/или соседние слова и дополнительная информация (например, является ли первая буква слова строчной или прописной). Таким образом, контекст может состоять как из одного слова, так и из нескольких с дополнительной информацией, такой как части речи. Как правило, множество пар класс-контекст много больше, чем размер корпуса, и $p(a|b)$ не может быть найдено напрямую из корпуса. В рамках решаемой задачи необходимо надежно оценить $p(a|b)$ для всех возможных контекстов и классов.

Принцип максимальной энтропии [3] утверждает, что наиболее правильным будет то распределение, которое согласуется с ограничениями и имеет наибольшую энтропию, т.е. максимизирует следующий функционал:

$$H(p) = -\sum_{x \in dom\, p} p(x) \log p(x)\, dx,$$

где *dom p* – множество всех возможных пар класс-контекст.

1.2.2. Модель максимальной энтропии

В модели максимальной энтропии [3] предлагается использовать большой лингвистический корпус для получения информации о соответствии контекстов и классов. Используя размеченный корпус, можно получить множество примеров класс-контекст и на его основе построить распределение $p^*(a,b)$.

Для применения принципа максимальной энтропии необходимо определить, что понимается под ограничениями распределения. Для определения ограничения в модели максимальной энтропии вводятся функции-свойства (*feature functions*) – функции из множества пар класс-контекст во множество *{0,1}*. Каждая введенная функция-свойство

соответствует наблюдаемому в корпусе свойству, которое необходимо сохранить в результирующем распределении.

Непосредственно ограничениями на распределение $p(a|b)$ являются равенства вида:

$$E_p f_k = E_{p^*} f_k,$$

где f_k – функция-свойство, E_p – математическое ожидание функции-свойства по результирующему распределению $p(a|b)p^*(b)$, E_{p^*} – математическое ожидание функции свойства по наблюдаемому распределению $p^*(a,b)$.

Таким образом, модель максимальной энтропии стремится к нахождению распределения $p(a|b)$, которое максимизирует энтропию и при этом удовлетворяет ограничениям на математические ожидания для всех функций-свойств.

Функции-свойства, как правило, выражают ограничения на возможные классы для данных контекстов. К примеру, в задаче определения части речи, где классы – части речи, а контексты – сами слова, одной из функций-свойств может быть следующая:

$$f(a,b) = \left\{ \begin{array}{l} 1, \ \textit{если } a = \textit{СОЮЗ и } b \in \{\textit{И}, \textit{А}, \textit{ИЛИ}, \textit{НО}, \ldots\} \\ 0, \ \textit{иначе} \end{array} \right\}.$$

Свойства распределений максимальной энтропии

Определим следующие множества распределений:

- $P = \{p \mid E_p f_k = E_{p^*} f_k \textit{ для всех } k\}$ – множество распределений, удовлетворяющих ограничениям;

- $Q = \{p \mid p(x) = \pi \prod_k \alpha_k^{f_k(x)}\}$ – множество распределений заданного вида.

Тогда для заданных множеств выполняются следующие утверждения [3]:

1. Если $p' \in P \cap Q$, то $p' = \arg\max_{p \in P} H(p)$ и p' – единственное;
2. Если $p' \in P \cap Q$, то $p' = \arg\max_{q \in Q} L(q)$ и p' – единственное, где

$$L(q) = \sum_x p^*(x) \log q(x)$$

Из данных утверждений следует важное следствие – искомое распределение следует искать в виде:

$$p(x) = \pi \prod_k \alpha_k^{f_k(x)}.$$

Параметры α_k находятся из уравнений:

$$E_p f_k = E_{p^*} f_k.$$

Найденное распределение будет максимизировать энтропию и удовлетворять всем ограничениям.

Также по второму утверждению найденное распределение будет максимизировать *L(q)*, т.е. будет наиболее вероятным с точки зрения полученных из лингвистического корпуса данных.

1.2.3. Алгоритм обучения и вывода

Алгоритм обучения

Для поиска параметров α_k используется алгоритм *GIS – Generalized Iterative Scaling* [4].

Алгоритм находит неизвестные параметры путем итераций:

$$\alpha_k^{(0)} = 1$$

$$\alpha_k^{(n+1)} = \alpha_k^{(n)} \left(\frac{E_{p^*} f_k}{E^{(n)} f_k} \right)^{\frac{1}{C}}$$

$$E^{(n)} f_k = \sum_x p^{(n)}(x) f_k(x)$$

$$p^{(n)}(x) = \pi \prod_k (\alpha_k^{(n)})^{f_k(x)}$$

Заметим, что для вычисления $E_{p^*} f$ и $E^{(n)} f_k$ по формулам выше требуется суммирование по всем парам класс-контекст.

В случае вычисления $E_{p^*} f$ фактически требуется суммировать по парам класс-контекст из корпуса, т.к. усреднение берется по наблюдаемому распределению *p*(x)*:

$$E_{p^*} f_k = \frac{1}{N} \sum_{i=1}^{N} f_k(a_i, b_i).$$

В случае с вычислением $E^{(n)} f_k$ суммирование по всем парам класс-контекст в большинстве случаев невозможно и необходимо вычислять $E^{(n)} f_k$ приближенно, суммируя по всем встреченным в корпусе парам класс-контекст:

$$E^{(n)} f_k \approx \sum_{i=1}^{N} p^*(b_i) \sum_a p^{(n)}(a \mid b_i) f_k(a, b_i).$$

Итерации алгоритма останавливают, когда их число превосходит заданное или когда параметры α_k изменяются на незначительную величину.

Алгоритм вывода

Алгоритм обучения находит параметры α_k и, как следствие, распределение $p(a|b)$ вида:

$$p(a|b) = \pi \prod_k \alpha_k^{f_k(a,b)} .$$

Используя данное распределение, можно оценивать вероятность данного класса *a* для данного контекста *b*.

Рассмотрим, как модель максимальной энтропии используется в задачах обработки текста.

Многие задачи обработки текста можно сформулировать в виде задачи расстановки меток на словах предложения. К примеру, в задаче определения частей речи требуется расставить метки – части речи – для каждого слова в предложении. В задаче определения имен собственных требуется расставить над каждым словом по метке – является ли слово именем собственным или нарицательным.

Для решения задач расстановки меток можно использовать модель максимальной энтропии – классами будут все возможные метки, контекстами – та информация, которая используется для расстановки меток.

Заметим, что во многих задачах расстановки меток для достижения хороших результатов недостаточно лишь только информации о текущем обрабатываемом слове. Так, например, в задаче расстановки частей речи для высокого качества выдаваемых результатов кроме непосредственно информации о слове в контекст необходимо также включить части речи нескольких предыдущих слов.

Таким образом, задачу расстановки меток в общем случае нельзя решать выбором наилучшего ответа для каждого слова в отдельности.

Вместо этого необходимо сразу искать наилучшую последовательность меток для всех слов предложения.

Более формально: согласно найденному распределению $p(a|b)$ последовательность меток $a_1, a_2, ...a_n$ для данной последовательности контекстов $b_1, b_2, ...b_n$ имеет вероятность:

$$p(a_1 ...a_n | b_1 ...b_n) = \prod_i p(a_i | b_i) .$$

Тогда задача расстановки меток сводится к задаче поиска наиболее вероятной последовательности меток $a_1, a_2, ...a_n$ для данной последовательности контекстов $b_1, b_2, ...b_n$.

Для решения данной задачи можно использовать алгоритм *Beam Search*.

Алгоритм *Beam Search* относится к эвристическим алгоритмам поиска и не всегда выдает оптимальный результат, однако, на практике получаемые результаты близки к оптимальным.

Алгоритм проводит серию итераций, поддерживая в куче множество «самых лучших» на данный момент последовательностей меток. Алгоритм начинает работу с единственной пустой последовательности, находящейся в куче. На каждой итерации алгоритм выбирает некоторое константное число (*beam size*) лучших последовательностей и формирует новую кучу. Новая куча последовательностей формируется из отобранных последовательностей путем приписывания к ним всех возможных меток для следующего слова.

В результате на последней итерации алгоритм строит кучу из полных последовательностей меток. Данное множество последовательностей не обязательно содержит в себе оптимальную последовательность, однако, как правило, в нем есть последовательности

близкие к оптимальной. Алгоритм завершает работу, возвращая лучшую последовательность, находящуюся в куче.

1.3. Графические вероятностные модели

1.3.1. О графических моделях

Графические модели [5] являются мощным инструментом, активно используемым в машинном обучении.

Многие задачи машинного обучения можно переформулировать на языке теории вероятностей. Известным и неизвестным величинам ставятся в соответствие случайные переменные $(X_1, X_2, ..., X_N)$, а главной целью обучения является нахождение *совместного распределения* наблюдаемых и скрытых случайных величин $p(X_1, X_2, ..., X_N)$.

Таким образом, многие задачи машинного обучения сводятся к нахождению и работе с распределениями случайных величин.

Распределение случайных величин в общем случае является сложным для работы объектом. Если нет каких-либо предположений о распределении, то число независимых параметров распределения растет экспоненциально с увеличением количества переменных. Как следствие, для реальных задач, где число переменных может исчисляться десятками, сотнями или даже тысячами, хранить и работать с распределением напрямую невозможно.

Графические модели являются инструментом для компактного представления и работы с распределениями случайных величин. Главная идея графических моделей – представление распределения в виде параметризованного графа.

1.3.2. Модель CRF

Модель CRF [6] является разновидностью графической модели, основанной на ненаправленном графе. Ненаправленные графические модели также называются марковскими сетями (*Markov Random Fields*).

Марковские сети

Марковская сеть [5] позволяет компактно хранить распределение в виде графа. В общем случае распределение N случайных величин требует задания экспоненциального числа параметров. Чтобы иметь возможность компактно хранить распределение и работать с ним, необходимо принять ряд утверждений об условной независимости или, что эквивалентно, о факторизации распределения. Данные предположения естественным образом представимы в виде неориентированного графа.

Рассмотрим некоторое совместное распределение N случайных величин $P(X_1, X_2, ..., X_N)$. Для простоты будем считать, что каждая переменная принимает конечное число значений. Графические модели также можно использовать и в случае непрерывных случайных величин.

Предположим, что данное распределение можно представить в виде произведения нескольких множителей (*факторов*):

$$P(X_1, ..., X_N) = \frac{1}{Z} \prod_k \Psi_k(X_{A(k)}),$$

где $X_{A(k)}$ – множество переменных фактора Ψ_k, Z – константа нормализации:

$$Z = \sum_{X1 \ldots XN} \prod_k \Psi_k(X_{A(k)}).$$

По факторизации распределения можно построить неориентированный граф – вершины графа соответствуют случайным

переменным, ребро между вершинами *Xi* и *Xj* проводится тогда и только тогда, когда существует фактор, в который входят *Xi* и *Xj*.

Получившийся граф называется марковской сетью.

Возможно также обратное построение – по марковской сети можно построить соответствующее ей распределение. Для этого необходимо задать распределение как произведение факторов (деленное на константу нормализации), причем необходимо выполнение следующего условия – для каждого фактора, переменные, входящие в него, должны составлять в графе полный подграф, т.е. клику.

Существует несколько важных свойств, связывающих марковские сети и соответствующие им распределения. Главное свойство заключается в том, что марковская сеть задает множество условных независимостей, которые должны выполняться в любом распределении, которое факторизуется по данной сети. Для строго положительных распределений верно и обратное – если данное строго положительное распределение удовлетворяет всем свойствам условной независимости марковской сети, то данное распределение факторизуется по данной сети. Данный факт является содержанием *теоремы Хаммерслей-Клиффорда* [5] (*Hammersley-Clifford theorem*).

Определим более формально множество условных независимостей марковской сети, задаваемой графом *G*. Будем говорить, что марковская сеть задает свойство условной независимости для данных трех непересекающихся множеств случайных величин *A*, *B*, *C*, обозначаемое

$$I_G(A, B \mid C),$$

если в графе *G* все пути из множества переменных *A* во множество переменных *B* содержат хотя бы одну вершину из множества переменных *C*.

Будем говорить, что данное распределение P удовлетворяет свойству условной независимости для непересекающихся множеств случайных величин A, B, C, и обозначать:

$$I_P(A, B | C),$$

если верно следующее утверждение:

$$P(A, B | C) = P(A | C) P(B | C).$$

Тогда верны следующие два утверждения, связывающие марковские сети и распределения:

1. Если данное распределение P факторизуется по данной марковской сети, задаваемой графом G, то для любых непересекающихся множеств случайных переменных A, B, C верно следующее утверждение:

$$I_G(A, B|C) \Rightarrow I_P(A, B|C)$$

2. (Теорема Хаммерслей-Клиффорда [5]). Если для данного строго положительного определенного распределения P и для данной марковской сети, задаваемой графом G для любых непересекающихся множеств случайных переменных A, B, C верно следующее утверждение:

$$I_G(A, B|C) \Rightarrow I_P(A, B|C),$$

то данное распределение P факторизуется по данной марковской сети G.

Из данных двух утверждений следует, что марковская сеть задает класс положительно определенных распределений, удовлетворяющих свойствам условной независимости, налагаемым графом, и что данный класс распределений – это в точности все распределения, которые факторизуются по графу марковской сети.

Графическая модель CRF

Графическая модель CRF [6] во многом схожа с классическими марковскими сетями. Однако между данными моделями есть принципиальное отличие.

Многие примеры использования графических моделей сводятся к следующему сценарию – случайные переменные модели разделены на два множества: X – множество наблюдаемых случайных величин, Y – множество скрытых случайных величин; требуется, зная значения переменных X, найти апостериорное распределение $P(Y \mid X)$. Чаще всего, апостериорное распределение используется для нахождения MAP-решения (*MAP-solution*):

$$y^* = \arg\max_y P(y \mid X).$$

В качестве примера рассмотрим задачу определения частей речи в предложении. В данной задаче переменные X соответствуют словам (или некоторым признакам слов, таким как, например, является ли первая буква слова прописной или строчной), а переменные Y – неизвестным частям речи.

По тому, как модель используется для решения данной задачи, модели можно разделить на два класса – порождающие (*generative*) и дискриминирующие (*discriminative*). Основное отличие между данными моделями – в распределениях, которые они задают.

Порождающие модели моделирует совместное распределение переменных X и Y: $P(X, Y)$.

Дискриминирующие модели моделируют непосредственно апостериорное распределение $P(Y \mid X)$.

Как порождающие, так и дискриминационные модели имеют преимущества друг перед другом.

Порождающие модели можно использовать для семплирования из совместного распределения $P(X,Y)$. Также при использовании порождающих моделей возможно обучение на неразмеченных данных, т.е. на данных, где отсутствуют значения переменных Y.

В то же время, главным недостатком порождающих моделей является необходимость моделировать распределение $P(X)$, т.е. распределение на входах. В ряде задач такое распределение может быть крайне сложным. Так, в задаче определения частей речи $p(X)$ – это распределение на последовательности слов предложения. Данное распределение, очевидно, имеет очень сложную структуру. Если попытаться выразить все зависимости данного распределения, то с большой вероятностью получится очень сложная модель, которую будет невозможно использовать.

В результате, при использовании порождающего подхода ничего не остается, кроме как вносить в модель дополнительные утверждения об условной независимости в распределении $P(X)$, которые не соответствуют действительности. Так, например, в модели наивного байесовского классификатора делаются серьезные допущения о природе входных данных – случайные переменные X считаются независимыми при известном значении переменной Y. Хотя на практике наивный байесовский классификатор работает достаточно хорошо, данные действия могут привести к ухудшению результатов моделирования.

Дискриминирующие модели, напротив, лишены данного недостатка, т.к. они принципиально не моделируют распределение на входах, а сразу моделируют условное распределение $P(Y|X)$.

Графическая модель CRF задается неориентированным графом и моделирует распределение *P(Y|X)*, т.е. является дискриминирующей моделью.

Более формально, пусть задан граф *G* с вершинами *Y*. Тогда соответствующая модель CRF моделирует распределение *P(Y|X)*, если для любых значений переменных *X* распределение *P(Y|X)* факторизуется по графу *G*:

$$P(Y \mid X) = \frac{1}{Z(X)} \prod_{\Psi_A \in G} \Psi_A(y_A, x_A \mid \theta)$$

$$\Psi_A = \exp\left\{ \sum_{k=1}^{K(A)} \theta_{ak} f_{ak}(y_a, x_a) \right\},$$

где Ψ_A – факторы, f_{ak} – функции-признаки.

Функции-признаки обычно выражают признаки слов, такие как капитализация букв, часть речи, префиксы-суффиксы и др.

Во многих задачах обучения несколько факторов используют одни и те же веса θ. Чтобы выразить совместное использование одних и тех же весов, множество факторов *C* разбивается на подмножества $\{C_1, C_2, ..., C_P\}$, называемые шаблонами факторов (*clique templates*). Тогда модель CRF может быть записана в виде:

$$P(Y \mid X) = \frac{1}{Z(X)} \prod_{C_P \in C} \prod_{\Psi_C \in C_P} \Psi_C(x_C, y_C; \theta_p)$$

$$\Psi_C(x_C, y_C; \theta_p) = \exp\left\{ \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(x_C, y_C) \right\}$$

$$Z(X) = \sum_{y} \prod_{C_p \in C} \prod_{\Psi_C \in C_P} \Psi_C(x_C, y_C; \theta_p).$$

1.3.3. Алгоритмы обучения и вывода

Известно, что задача вывода в графических моделях NP-трудна [7], поэтому на данный момент не известно эффективных точных детерминированных алгоритмов вывода.

Существующие алгоритмы вывода можно разделить на несколько типов. Точные алгоритмы, такие как *алгоритм Junction Tree* [8], позволяют найти точное MAP-решение, однако, данные алгоритмы в общем случае могут работать экспоненциальное время. На практике для конкретной модели такие алгоритмы могут быть применимы.

Вторая группа алгоритмов – алгоритмы приближенного вывода. К этой группе можно отнести MCMC (*Markov Chain Monte Carlo*) [9] алгоритмы – вероятностные алгоритмы вывода, основанные на семплировании. Пример из этой группы алгоритмов – алгоритм, основанный на *семплировании по Гиббсу* (*Gibb's Sampling*) [9]. Другая группа алгоритмов приближенного вывода – алгоритмы вариационного вывода (*Variational Inference*) [10]. Также к алгоритмам приближенного вывода относятся алгоритмы передачи сообщений (*message-passing algorithms*) такие как алгоритм *Belief Propagation* [11].

Обучение модели CRF производится путем максимизации функции-правдоподобия (*likelihood*).

Рассмотрим обучающее множество – множество пар $\{(X_i, Y_i)\}$. Тогда цель обучающего алгоритма – найти веса θ, которые максимизируют функцию правдоподобия:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{N} P(Y_i \mid X_i; \theta) .$$

Для простоты вывода обычно максимизируют логарифм функции правдоподобия (*log-likelihood*):

$$\theta^* = \arg\max_{\theta} l(\theta) = \arg\max_{\theta} \sum_{i=1}^{N} \log P(Y_i \mid X_i; \theta) .$$

Данная задача является классической оптимизационной задачей и решается с помощью общих алгоритмов оптимизации. На практике применяется алгоритм L-BFGS [12]. При этом производные $l(\theta)$ по весам θ вычисляются по следующим формулам:

$$\frac{\partial l}{\partial \theta_{pk}} = \sum_i \left\{ \sum_{\Psi_C \in C_P} f_{pk}(x_C^{(i)}, y_C^{(i)}) - \sum_{\Psi_C \in C_P} \sum_{y'_C} f_{pk}(x_C^{(i)}, y'_C) p(y'_C \mid x^{(i)}) \right\}$$

Выводы по главе 1

1. Рассмотрены основанные на применении методов машинного обучения подходы к решению задач обработки текстов на естественном языке.

2. Рассмотрены две модели машинного обучения – модель максимальной энтропии и модель CRF, активно применяемые в области обработки текстов на естественных языках.

ГЛАВА 2. ЗАДАЧА ИЗВЛЕЧЕНИЯ ОТНОШЕНИЙ ИЗ ТЕКСТОВ

2.1. Постановка задачи

Задача извлечения отношений [13], рассматриваемая в данной работе, заключается в получении списка троек-отношений из текста на русском языке. Отношением в данном контексте называется тройка фраз, соответствующих действующему субъекту, фразе отношения и объекту отношения. Так, например, рассмотрим предложение:

«Президент сделал заявление о начале реформ»

Данное предложение содержит отношение:

(«Президент», «сделал заявление о», «начале проведения реформ»).

2.2. История задачи

Задача извлечения отношений из текстов традиционно рассматривалась для отношений узкой тематики и обычно решалась с помощью написанных вручную правил извлечения. Системы, основанные на данном принципе, применялись для обработки определенных классов текстов. Например, такие системы можно использовать для извлечения мест проведения различных мероприятий, при условии, что тексты объявлений не сильно отличаются друг от друга.

Данный подход имеет ряд недостатков. Системы, основанные на написанных вручную правилах способны работать только с текстами,

составленными по определенному образцу. Также данный подход плохо расширяется на поддержку других типов отношений.

Следующим шагов в развитии систем извлечения отношений стало использование машинного обучения для автоматического построения списка правил по размеченным вручную данным.

Данный подход уже позволяет системе обрабатывать сильно отличающиеся тексты. Однако данный подход требует ручной разметки данных для обучения и трудно расширяем на поддержку новых типов отношений.

Для преодоления данных недостатков была предложена [13] новая парадигма извлечения данных – открытое извлечение информации (*open information extraction*). В рамках данной парадигмы предлагается отказаться от систем, направленных на извлечение определенных типов отношений. Вместо этого предлагается создавать системы способные к извлечению широкого класса отношений, не ограниченного определенной темой. В рамках данного подхода подразумевается, что система извлечения будет делать один проход по тексту. Данные требования позволяют системам, следующим данной парадигме, обрабатывать большие объемы текстов и извлекать из них широкий класс отношений.

2.3. Применение систем извлечения отношений

Системы, основанные на парадигме открытого извлечения информации, хорошо подходят для обработки данных из Web, т.к. способны к обработке больших объемов текстов и извлечению информации различной тематики.

Представление фактов в виде отношений-троек лежит в основе рекомендованного W3C стандарта RDF [14] для представления данных.

Системы извлечения отношений используются в поисковых системах [15], а также при составлении *онтологий* – структурированных баз знаний.

2.4. Существующие подходы к решению

К системам, которые обучаются на размеченных данных, можно отнести *DIPRE* [16], *Snowball* [17].

К системам, основанным на парадигме открытого извлечения информации, относятся *KnowItAll* [18], *TextRunner* [13]. Перечисленные системы являются реализацией парадигмы открытого извлечения отношений для текстов на английском языке.

2.5. Компоненты системы извлечения отношений

Двумя основными компонентами, из которых состоит система извлечения отношений, являются компоненты «Chunker» и «Extractor» [13].

Перед тем, как извлекать отношения, текст разделяется на предложения, которые поступают в компонент «Chunker». Результат работы компонента «Chunker» используется как вход для компонента «Extractor». Результатом работы системы извлечения отношений является список отношений, найденных в исходном тексте.

2.5.1. Компонент «Chunker»

Задача компонента «Chunker» – подготовка предложения к обработке с помощью компонента «Extractor».

Компонент «Chunker» разбивает предложения на осмысленные фразы (*chunks*), соответствующие основным действующим сущностям предложения и их действиям.

Компонент выделяет два вида фраз – фразы, соответствующие существительным, (*noun phrases*) и глагольные фразы (*verb phrases*). Приведем пример работы компонента «Chunker». Рассмотрим предложение:

«Пушкин происходил из разветвленного дворянского рода»

Результатом работы компонента «Chunker» будут следующие фразы:

- «Пушкин» – фраза-существительное;
- «происходил» – фраза-глагол;
- «из разветвленного дворянского рода» – фраза-существительное.

2.5.2. Компонент «Extractor»

Компонент «Extractor» принимает на вход список фраз, полученный с помощью «Chunker», и выдает список найденных отношений.

В примере выше компонент «Chunker» сгенерировал следующий список фраз:

- «Пушкин»;
- «происходил»;
- «из разветвленного дворянского рода».

Результатом работы компонента «Extractor» на данном входе будет отношение («Пушкин», «происходил из», «разветвленного дворянского рода»). «Происходил из» в данном контексте является фразой отношения, связывающей фразы «Пушкин» и «разветвленного дворянского рода». Заметим, что фразы отношения не обязаны совпадать с фразами, которые возвращает компонент «Chunker».

Выводы по главе 2

1. В главе представлена постановка задачи, приведена история решения задачи и существующие подходы к решению.

2. В главе описана базовая схема системы извлечения отношений из текстов и описаны ее основные компоненты.

ГЛАВА 3. СИСТЕМА ИЗВЛЕЧЕНИЯ ОТНОШЕНИЙ ИЗ ТЕКСТОВ НА РУССКОМ ЯЗЫКЕ

3.1. Общая схема системы извлечения

Общая схема системы извлечения отношений показана на рис. 1.

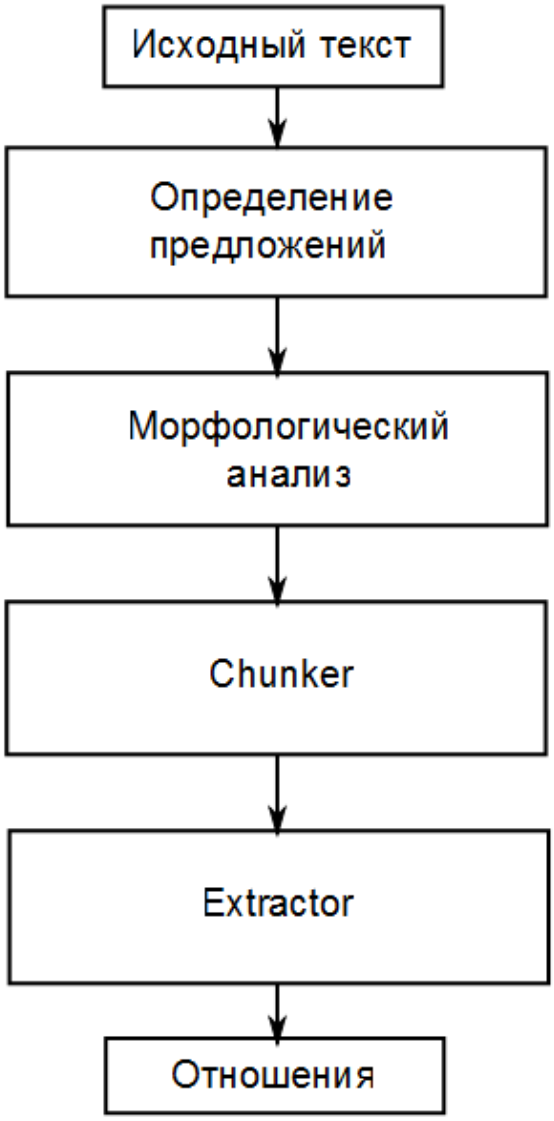

Рис. 1. Схема системы извлечения отношений

3.2. Предварительная обработка текста

3.2.1. Выделение предложений из текста

Обработка текста начинается на этапе разбиения текста на предложения. Заметим, что данная задача не сводится к определению в предложении точек. Действительно, точка не всегда означает конец предложения – точки используются также при указании инициалов, сокращений и др. Также в части предложений точки могут быть пропущены или вместо точки может быть восклицательный, вопросительный знак или их комбинация. Кроме того, предложения могут включать в себя другие предложения и др. Все это требует дополнительных проверок при определении границ предложений.

3.2.2. Морфологический анализ слов текста

Следующим этапом в обработке текста является этап морфологического анализа. На данном этапе для каждого из слов предложения определяются его часть речи, падеж, род, число.

3.3. Получение данных для обучения

3.3.1. Лингвистические корпусы

Для обучения компонентов «Chunker» и «Extractor» необходимы размеченные данные. В качестве источников размеченных предложений, как правило, используют лингвистические корпусы.

Лингвистическим корпусом называется собрание размеченных текстов на определенном языке. Существует множество корпусов для различных языков ([19], [20]). Для обучения компонентов «Chunker» и «Extractor» использовался корпус русского языка [21], состоящий из предложений с указанной синтаксической структурой (деревом). Пример предложения с указанной синтаксической структурой приведен на рис. 2.

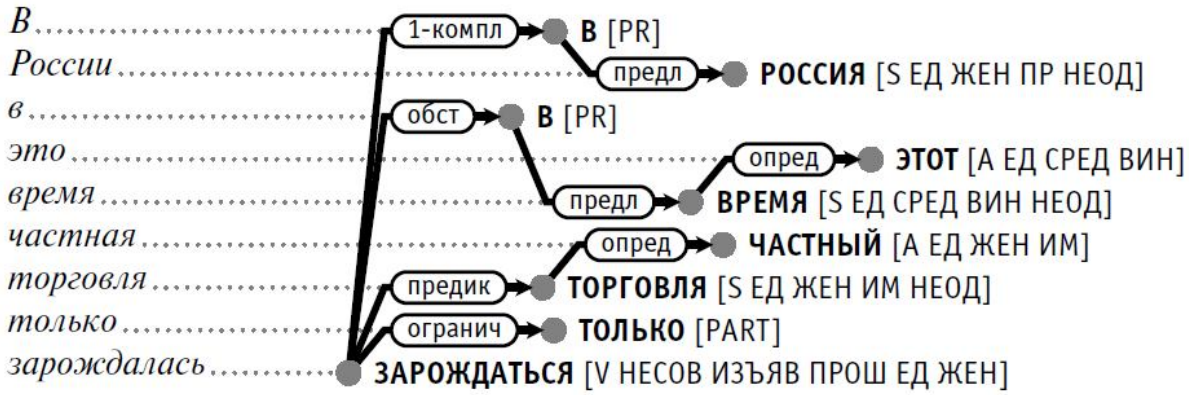

Рис.2. Пример предложения из корпуса

Синтаксические деревья, ассоциированные с каждым предложением в корпусе, содержат информацию о словах и зависимостях между ними. Вершины дерева соответствуют словам. В каждой вершине указываются часть речи и, если слово имеет данные характеристики, падеж, род, число и др. В таблице 1 приведены обозначения, используемые в корпусе. Каждому ребру в дереве соответствует зависимость между словами. Вид зависимости указывается на ребре, ведущем от одного слова к другому. В таблице 2 указаны некоторые виды зависимостей, используемые в корпусе.

Основными зависимостями, которые можно использовать для обнаружения отношений в предложениях, являются зависимости «подлежащее-сказуемое», а также зависимость-дополнение. Анализируя синтаксические деревья корпуса, можно выделить набор отношений, который можно использовать для обучения моделей. В следующем пункте будут рассмотрены способы извлечения отношений из синтаксических деревьев.

Обозначение	Часть речи
S	Существительное, местоимение
A	Прилагательное
V	Глагол
ADV	Наречие
NUM	Числительное
PR	Предлог
COM	Композит
CONJ	Союз
PART	Частица
P	Слово-предложение
INJ	Междометие
NID	Заимствование

Таблица 1. Обозначения в корпусе

Отношение	Значение
предикат	сказуемое-подлежащее
агент	глагол-субъект действия
квазиагент	глагол-субъект действия
N-компл	предикат-дополнение
предл	предлог-предложная группа
опред	Определение

Таблица 2. Некоторые виды зависимостей в корпусе

Заметим, что в корпусе местоимения не рассматриваются как отдельная часть речи. Для обозначения местоимений используется часть речи существительное.

Также в данном корпусе помимо шести падежей (именительный, родительный, дательный, винительный, творительный, предложный) используются дополнительные три – партитивный («дайте *чаю*»), местный («в *лесу*», «на *снегу*»), звательный («*отче*», «*Вань*»).

3.3.2 Извлечение отношений из корпуса

Для обучения компонентов «Chunker», «Extractor» необходимо построить обучающие множества примеров.

Обучение компонента «Chunker»

Обучающее множество примеров для компонента «Chunker» состоит из предложений с указанными фразами-существительными и фразами-глаголами. Рассмотрим, как по синтаксическим деревьям можно извлечь информацию о фразах в предложении.

Каждая фраза содержит в себе основное слово. Для определения фраз-глаголов предлагается сначала найти в предложении возможные основные слова. Для этого для каждого слова проверяется, что оно является глаголом или прилагательным, а также участвует в отношении «предикат» с другим словом. Для получения остальных слов фразы используется структура дерева предложения, а именно рассматриваются достижимые по ребрам слова. При этом используются только ребра, соответствующие отношениям «дополнение», «прилагательное». Особое внимание уделяется найденным предложным фразам, т.к. такие конструкции часто входят в найденные отношения.

При поиске фраз-существительных предлагается находить возможные основные слова, помеченные как S – существительное или местоимение, NID – заимствование, а также имеющие именительный или винительный падежи. Также при поиске основных слов проверяется, что слово входит в отношение «предикат» с другим словом. Для получения остальных слов фразы необходимо найти слова, достижимые из основного слова по ребрам «дополнение», «агент», «определение».

Обучение компонента «Extractor»

Обучающее множество для компонента «Extractor» состоит из найденных в корпусе отношений. Для поиска отношений сначала из предложений корпуса извлекаются фразы. Затем найденные фразы-существительные и фразы-глаголы объединяются в отношения.

Для получения списка отношений данного предложения предлагается рассмотреть каждую фразу-глагол и определить, входит ли данная фраза в отношение. Для этого рассматриваются фразы-существительные, находящиеся слева и справа от данной фразы-глагола и определяются возможные аргументы отношения. Синтаксическая структура предложения помогает определить, есть ли связь между данной фразой-существительным и текущей фразой-глаголом – необходимо рассмотреть ребра дерева и найти путь от основного слова одной фразы к основному слову другой. При этом на пути от левой фразы-существительного к фразе-глаголу должно быть ребро с пометкой «предикат». Данное условие помогает отсечь множество неправильных отношений, т.к. проверяет, что существительное и глагол являются соответственно подлежащим и сказуемым. Для поиска оставшегося аргумента отношения предлагается рассмотреть путь от глагола до

основного слова фразы-существительного, находящегося справа от глагола. Рассматривается кратчайший путь. При этом проверяется, чтобы путь содержал ребра, соответствующие отношениям «дополнение», «обстоятельство», «сравнение» и др. Данная проверка позволяет оставить только те фразы, которые наиболее вероятно составляют отношение.

После нахождения фразы-глагола и двух фраз-существительных рассматривается путь, соединяющий основные слова данных фраз. В качестве слов отношения используются слова, находящиеся на данном пути. Кроме того, в отношения включаются слова-модификаторы, такие как, например, частица «не», влияющие на смысл отношения.

Рассмотрим извлечение отношений на примере синтаксического дерева, изображенного на рис. 3.

Рис. 3. Пример синтаксического дерева

В данном примере в качестве фразы-глагола можно выбрать слово «составляет». Также в предложении находятся три фразы:

- «В цене энергетической установки на топливном элементе»;
- «стоимость процессора»;
- «примерно треть».

Субъектом отношения является «стоимость процессора», объектом отношения является «примерно треть».

Пример синтаксического дерева с нетривиальной глагольной фразой изображен на рис. 4.

Рис. 4. Пример синтаксического дерева

Глагольной фразой предложения является «должен быть узнаваем».

Другими фразами предложения являются:

- «всякий хороший писатель»;
- «на уровне фразы».

Извлеченное отношение:

(«всякий хороший писатель»,«должен быть узнаваем»,«на уровне фразы»).

Выводы по главе 3

1. В главе приведена схема работы системы извлечения отношений, показаны основные этапы извлечения отношений.

2. В главе рассмотрены лингвистические корпусы и обучение компонентов «Chunker» и «Extractor».

ГЛАВА 4. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

4.1. Схема экспериментов

В данной работе был проведен ряд экспериментов для определения наиболее эффективного способа извлечения отношений.

На первом этапе было проведено обучение компонента «Chunker» на множестве примеров, полученных из лингвистического корпуса. В качестве моделей обучения были выбраны наиболее эффективные модели в области обработки текстов – модель CRF и модель максимальной энтропии.

На втором этапе было проведено обучение компонента «Extractor». Проведено сравнение двух подходов к извлечению отношений – подхода, основанного на эвристическом алгоритме, и подхода, основанного на наивном байесовском классификаторе.

4.2. Метод оценки результатов экспериментов

Для оценки результатов работы компонента «Chunker» 10% от извлеченных примеров были выделены в тестовое множество.

Для оценки результатов работы компонента «Extractor» были использованы размеченные вручную тексты.

В качестве оценок эффективности работы системы извлечения информации используются две величины – точность (*precision*) и полнота (*recall*).

Точность – доля правильно найденных отношений среди всех найденных.

Полнота – доля правильно найденных отношений среди всех правильных отношений.

Оценки точности и полноты объединяются в оценке *F-Measure*:

$$F = 2\frac{precision \;\; recall}{precision + recall}$$

Оценка принимает значения от 0 до 1.

В качестве меры различия двух множеств можно использовать расстояние Жаккара (*Jaccard Distance*):

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

4.3. Обучение компонента «Chunker»

Для обучения компонента «Chunker» были использованы модели CRF и модель максимальной энтропии.

Модели обучались на множестве обучающих примеров. В таблице 3 приведен пример из обучающего множества.

Мы **S B-NP**
придумали **V B-VP**
Постоянно **ADV B-NP**
действующий **V I-NP**
источник **S I-NP**
Горючей **A I-NP**
смеси **S I-NP**
и **CONJ I-NP**
Зажигалку **S I-NP**

Таблица 3. Пример из обучающего множества

Каждый пример в обучающем множестве состоит из слов предложения с дописанными частями речи и меткой.

Метка принимает одно из следующих значений:

- B-NP – начало новой фразы-существительного;
- I-NP – слово внутри фразы-существительного;
- V-NP – начало новой глагольной фразы;
- I-NP – слово внутри глагольной фразы;
- O – слово не входит ни в одну фразу.

Таким образом, каждый обучающий пример состоит из предложения и последовательности меток на словах, обозначающей разбиение слов на фразы.

Цель обучения – получить модель (CRF или максимальной энтропии), максимально правильно расставляющую метки и словах и, тем самым, получающую фразы предложения.

4.3.1. Обучение с использованием модели максимальной энтропии

Для обучения использовался алгоритм *GIS*, для вывода меток на новых предложениях – алгоритм *Beam Search* (с параметром *beam* = 10).

На рис. 5 приведен график зависимости значения F-Measure от числа итераций алгоритма GIS.

Из графика видно, что наиболее оптимальным значением числа итераций является 40.

Полученные значения точности, полноты и F-Measure:

Оценка	Значение
Точность	0.7591
Полнота	0.7888
F-Measure	0.7737

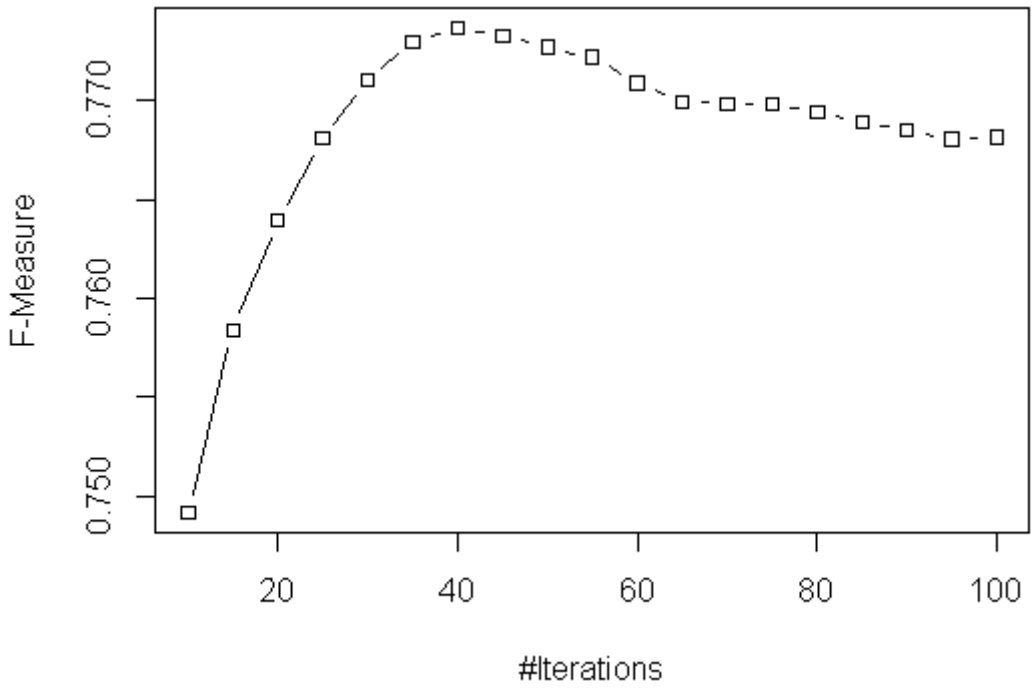

Рис 5. График зависимости F-Measure от числа итераций

4.3.2. Обучение с использованием модели CRF

Для обучения компонента «Chunker» использовалась модель CRF и алгоритм L-BFGS.

Полученные значения точности, полноты, F-Measure:

Оценка	Значение
Точность:	0.6886
Полнота:	0.7316
F-Measure:	0.7315

4.4. Построение компонента «Extractor»

Для обучения компонента «Extractor» использовалось обучающее множество из 30 тысяч размеченных предложений. На рис. 6 приведен пример из обучающего множества.

Деньги **S O**
на **PR O**
программы **S O**
здравоохранения **S O**
изымаются **V IN**
в частности **ADV O**
из **PR IN**
местного **A O**
бюджета **S O**

Рис. 6. Пример из обучающего множества

Пример из обучающего множества состоит из слов предложения, частей речи и меток. Метка принимает одно из следующих значений:

- IN – слово входит в глагольную фразу отношения;
- O – слово не входит в глагольную фразу отношения.

В примере видно, что в глагольную фразу отношения входят слова «изымаются из». Заметим, что слова в глагольной фразе не обязаны идти в предложении друг за другом.

Полученные значения точности, полноты и F-Measure для модели максимальной энтропии:

Оценка	Значение
Точность:	0.9039
Полнота:	0.9362
F-Measure:	0.9198

Компонент «Extractor» использует построенную модель для извлечения центральной части отношения – глагольной фразы.

Для получения последовательностей слов, которые затем будут переданы построенной модели, можно использовать несколько подходов.

Первый из них – использование эвристического алгоритма – алгоритма, основанного на написанных вручную правилах. Второй подход – использование алгоритма, основанного на классификаторе.

4.4.1. Использование эвристических алгоритмов

Для определения последовательностей слов, подаваемых на вход построенной модели, использовались фразы, полученные с помощью компонента «Chunker».

Получая на вход фразы, эвристический алгоритм находит три подряд идущие фразы, проверяет, что первая и третья являются фразами-существительными, вторая – глагольной фразой. Затем алгоритм делает серию проверок, по результатам которой отвергает данную последовательность слов или передает ее на вход построенной модели. К проверкам, которые делает алгоритм, относятся:

- содержит ли центральная фраза глагол;
- проверка длин фраз-существительных;
- проверка длины глагольной фразы;
- содержит ли первая фраза слово в им. падеже;
- согласованность глагола и существительного по числу;

На тестовых текстах получены следующие значения точности, полноты и F-Measure:

Оценка	Значение
Точность:	0.7353
Полнота:	0.1426
F-Measure:	0.2387

4.4.2. Использование классификатора

Основным недостатком эвристического алгоритма является трудность добавления новых характеристик (*features*) для проверки.

Данную проблему можно решить, используя алгоритм классификации последовательностей фраз на правильные и неправильные. В качестве классификатора в работе использовался наивный байесовский классификатор [2], использующий следующие характеристики последовательностей фраз:

- наличие местоимений в первой фразе;
- наличие глагола во второй фразе;
- длины фраз:
- наличие слова в им. падеже в первой или третьей фразах;
- наличие предлога в третьей фразе;
- согласованность первой и второй фраз по числу;
- наличие слов между фразами.

На тестовых текстах были получены следующие значения точности, полноты и F-Measure:

Оценка	Без проверки согласованности по числу и последовательности	Без проверки согласованности по числу	Все проверки
Точность	0.5455	0.5915	0.623
Полнота	0.2393	0.2393	0.3248
F-Measure	0.3327	0.3408	0.427

По полученным данным видно, что добавление новых характеристик увеличивает точность и полноту.

Выводы по главе 4

1. В главе описана схема экспериментов построения моделей компонентов «Chunker» и «Extractor», приведены полученные значения оценок.

2. В главе описаны алгоритмы получения слов для обработки, проведено сравнение эвристического алгоритма и байесовского классификатора.

ЗАКЛЮЧЕНИЕ

В данной работе была рассмотрена задача построения системы извлечения отношений из текстов на русском языке. Были рассмотрены различные алгоритмы построения системы и ее частей – компонентов «Chunker» и «Extractor».

Для построения компонента «Chunker» были рассмотрены две модели – графическая модель CRF и модель максимальной энтропии, и показано, что для исследуемой задачи модель максимальной энтропии дает лучшие результаты.

Для построения компонента «Extractor» были рассмотрены эвристические алгоритмы и алгоритмы, основанные на классификации, и показано, что применение наивного байесовского классификатора приводит к более высоким значениям оценки F-Measure.

Система извлечения отношений, построенная в данной работе, может быть использована при создании приложений, анализирующих тексты, поисковых машин. Также подобная система может быть использована при создании онтологий.

ИСТОЧНИКИ

1. C. D. Manning, H.Schuetze «Foundations of Statistical Natural Language Processing», The MIT Press
2. T. M. Mitchell «Machine Learning», McGraw-Hill Science
3. A. Ratnaparkhi «A Simple Introduction to Maximum Entropy Models for Natural Language Processing»
4. J. N. Darroch, D. Ratcliff «Generalized Iterative Scaling for Log-Linear Models», Ann. Math. Statist. Volume 43, Number 5 (1972), 1470-1480
5. D. Koller, N. Friedman «Probabilistic Graphical Models: Principles and Techniques», The MIT Press
6. J. Lafferty, A. McCallum, F. Pereira «Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data», ICML 2001
7. V. Chandrasekaran, N. Srebro, P. Harsha «Complexity of Inference in Graphical Models», UAI 2008
8. F. V. Jensen, S. L. Lauritzen, K. G. Olesen «Bayesian updating in causal probabilistic networks by local computations», Computational Statistics Quarterly, 4:269-282
9. S. Geman, D. Geman, D. «Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images». IEEE Transactions on Pattern Analysis and Machine Intelligence, 6 (6): 721–741
10. C. Fox, S. Roberts «A Tutorial on Variational Bayes», Artificial Intelligence Review, 2011
11. F. R. Kschischang «Factor Graphs and the Sum-Product Algorithm», IEEE Transactions on Information Theory, vol. 47, No. 2

12. R. H. Byrd, P. Lu and J. Nocedal «A Limited Memory Algorithm for Bound Constrained Optimization», SIAM Journal on Scientific and Statistical Computing, 16, 5, pp. 1190–1208

13. M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni «Open Information Extraction from the Web», IJCAI 2007

14. W3C Resource Description Framework (http://www.w3.org/RDF/)

15. Review Miner (http://revminer.com/)

16. S. Brin «Extracting Patterns and Relations from the World Wide Web», EDBT 1998

17. E. Agichtein, L. Gravano «Snowball: Extracting relations from large plain-text collections», Proceedings of the Fifth ACM International Conference on Digital Libraries, 2000

18. O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, A. Yates «Unsupervised named-entity extraction from the web: An experimental study», Artificial Intelligence, 165(1), 2005

19. The Penn Treebank Project (http://www.cis.upenn.edu/~treebank/)

20. Czech National Corpus (http://ucnk.ff.cuni.cz/)

21. Russian National Corpus (http://ruscorpora.ru/)