САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-Механический факультет
Кафедра Системного Программирования

Дзендзик Дарья Анатольевна

Извлечение событий на основе автоматизированного построения линейных шаблонов

Дипломная работа студента 545 группы

Допустить к защите

Заведующий кафедрой:
д.ф. - м.н., профессор Терехов А. Н.

Научный руководитель:
д.ф. - м.н., профессор Новиков Б. А.

Рецензент:
инженер-исследователь, аспирант Ткаченко М. В.

Санкт-Петербург
2013 г.

SAINT-PETERSBURG STATE UNIVERSITY

Mathematics and Mechanics Faculty
Software Engineering Department

Daria Dzendzik

Event extraction based on automated building of linear patterns

Graduation Thesis

Admitted for defence

Head of Department:
Professor Andrey Terekhov

Scientific advisor:
Professor Boris Novikov

Reviewer:
Researcher engineer, PhD student Maksim Tkachenko

Saint-Petersburg
2013

Содержание

Введение

С каждым годом количество данных, а вместе с тем и количество информации, в том числе текстовой информации на естественном языке, во всём мире увеличивается с огромной скоростью[7, 3, 4]. Данный факт существенно повышает интерес к автоматической обработке информации и анализу текстов. Извлечение событий из текста является актуальной задачей, связанной с обработкой текстов на естественных языках.

Идея автоматизации тех или иных процессов близка и понятна современному человеку. Автоматическое извлечение информации, в частности - извлечение событий, значительно облегчит обработку документов и анализ текстов на естественном языке, представленных в неструктурированном виде.

Представленная работа является частью большого проекта по анализу неструктурированных данных, разрабатываемого российским подразделением лаборатории Hewlett-Packard. В рамках дипломнй работы будет рассматриваться анализ текстов на английском языке. Можно добавить, что английский язык находиться на третьем месте в мире среди естественных языков по числу говорящих на нём[21, 2, 31]. Кроме этого, более половины (55%) всех текстов в интернете написаны именно на английском языке[6].

Изначально механизмы извлечения событий применялись для поиска информации о террористических актах[1] и для обработки биомедицинской литературы[28, 26]. Несколько позднее возник интерес к автоматизированному анализу новостных лент и отслеживанию бизнес-событий. Подобная информация, как правило, представляется в неструктурированном виде. Даже частичная автоматизация процесса обработки неструктурированных текстов, в частности, новостных статей, поможет значительно уменьшить человеческие усилия, затраченные на выявление событий.

Одним из возможных вариантов обработки текстовой информации является извлечение событий при помощи правил, описывающих ту или иную структуру. В частности, могут быть использованы линейные правила: сопоставление текста с регулярным выражением. В линейных правилах все компоненты связаны друг с другом только порядком их следования. Извлечение событий при помощи таких правил проходит быстрее, чем при помощи правил, имеющих более сложную структуру, например, учитывающую синтаксические связи между индикатором и атрибтами[2]. Автоматическое построение правил - это ещё один шаг к оптимизации процесса извлечения событий из неструктурированного текста.

Данная работа может быть применена в задачах поиска любой информации, которая описывается особым образом. Мы используем описа-

[1] Message Understanding Conferences (MUC) -3(1991); -4(1992)

[2] Были проведены соответствующие эксперименты

ние событий в виде xml-документов. В нижеизложенном (раздел 1.1.6) определены границы текущего применения рассматриваемого процесса получения информации из текста.

Предложенный в данной работе алгоритм написан для среды TextMARKER (более подробно 1.1.3). Используются библиотеки UIMA и OpenNLP. Алгоритм реализован на платформе Java в среде NetBeans. Для разметки событий использовался инструмент для среды Eclipse UIMA Tool.

В результате был получен алгоритм построения линейных правил для извлечения событий в среде TextMARKER. Разработаны механизмы поиска потенциальных событий и выбора настоящих событий (раздел 3.1.1); разработаны операции обобщения линейных правил (раздел 3.2.5); размечен корпус новостных статей. Данный алгоритм был протестирован на небольшом наборе синтетических данных. Были проведены эксперименты на реальных новостных статьях и получены более высокие значения для точности, полноты и F1-меры по сравнению с правилами, написанными вручную человеком (раздел 5).

1 Задача извлечения событий

1.1 Предметная область и постановка задачи

1.1.1 Терминология

Прежде всего, отметим, что в данной работе два англоязычных термина - *правило* (англ. rule) и *шаблон* (англ. pattern) - выступают как абсолютные синонимы и имеют единое значение - правило.

В данном разделе уточним некоторые термины и понятия, которые встречаются в работе.

Событие - конструкция, описывающая изменение состояния объектов в реальном мире. Формально описывается индикатором события и набором атрибутов, которые встречаются в одном предложении и несут определённый для человека смысл. Например:

Google acquired YouTube – событие типа "Слияние и поглощение" (Merger&Acquisition);

HP announced new product – событие типа "Заявление" (Announced).

Атрибут - слово или набор слов в предложении, несущих определённый смысл и обладающих аннотацией определённого типа. Атрибуты делятся на обязательные и необязательные. Обязательные атрибуты входят в состав события как такового, и без них событие существовать не может. Дополнительные атрибуты уточняют обстоятельства и могут отсутствовать.

Аннотация - метка слова или набора слов, приписывающая его к некоторой группе слов. Например: "Google" будет иметь аннотации токена (отдельное слово), CW - слова с большой буквы и *Company* - является названием компании.

Индикатор события - слово или набор слов (обычно глагол или глагольная группа), принадлежащие к заранее определённому словарю. Индикатор идентифицирует событие определённого типа. Является обязательным атрибутом.

Корпус - набор документов, на котором может обучаться, тестироваться или проверяться алгоритм.

Слот - часть правила, заполняемая атрибутом.

1.1.2 Извлечение событий

Каждое из извлекаемых событий описывается индикатором и набором атрибутов. Наличие всех атрибутов и индикатора является необходимым, но не достаточным условием существования события. Встречаются предложения, которые содержат в себе индикатор и все атрибуты, необходимые для события, но не содержат в себе событие. Например: *Trade debt is about $10 million, a court paper said.* Приведённое предложение по набору атрибутов совпадает с предложением содержащим событие «Заявление» (Announced). Но на самом деле заявление сделала не компания,

как необходимо для события, а газета, которая указывает на то, что у этой компании есть долг.

Чтобы более точно определить наличие события можно использовать дополнительные средства анализа, такие, как синтаксический анализатор. Синтаксическая связь между словами фиксирует отношение, несущее определённый смысл. В данной работе использовался синтаксический анализатор Стенфордского университета[3].

Использование таких инструментов позволяет группировать предложения не только по типу событий, но и по содержанию синтаксических связей. Это помогает единовременно обрабатывать предложения, имеющие единую синтаксическую конструкцию.

1.1.3 Система TextMARKER

Разработанные в ходе нашей работы алгоритмы будут реализованы для системы TextMARKER[4][19]. Это инструмент с открытым исходным кодом для разработки приложений, извлекающих информацию на основе правил. Он полностью поддерживает систему типов платформы UIMA[5] и легко совместим с её компонентами.

1.1.4 Именованные сущности и правила

Система TextMarker, рассматриваемая в данной работе, использует правила, описанные в [19]. Они имеют вид регулярных выражений и оперируют специальными типами, называемыми именованными сущностями. Выделение именованных сущностей (named entity recognition) – это отдельная задача извлечения информации и в данной работе рассматриваться не будет.

Ниже приведён пример типа события, правила и предложения.

Событие “Announcemen“:

Company AnnouncementIndicator ->
GATHER(CompanyAnnouncement,1,2,“company“=1,“indicator“=2);

HP announces it will discontinue webOS development.

Событие “Acquisition“:

Company AcquisitionIndicator “by“ Company ->
GATHER(Acquisition,1,2,3,4,“acquirer“=4,“slot2“=1,“acquiree“=2);

BlindType has been acquired by Google.

Как видно из примера, правило состоит из двух частей: первая непосредственно отыскивает событие, вторая отмечает его атрибуты. Данные примеры построены человеком вручную, что является довольно трудоёмким процессом. Кроме того, при таком подходе построения правил

[3]http://nlp.stanford.edu/software/lex-parser.shtml
[4]http://tmwiki.informatik.uni-wuerzburg.de/
[5]http://uima.apache.org/

возникают следующие проблемы: нельзя гарантировать, что эти правила будут верны для всех событий, и не исключено существование событий, не покрытых сформированными правилами. Здесь появляется идея генерировать правила автоматически в виде регулярных выражений, которая будет основной в данной работе.

1.1.5 Цели и задачи

Исходя из вышеизложенного можно сформулировать цель данной работы и её задачи.

Цель работы: создать механизм генерации линейные правила в виде регулярных выражений для извлечения событий из неразмеченного корпуса документов.

Задачи:

1. разработать алгоритм построения правил;
2. разработать алгоритм подготовки данных;
3. реализовать механизм построения правил для системы TextMarker;
4. провести эксперименты, сравнить результаты извлечения событий с правилами, написанными вручную с правилами, построенными разработанным механизмом.

1.1.6 Условия

В данной работе будут рассматриваться три типа события и их основные атрибуты:

1. Слияние и поглощение “Merger&Acquisition“ (“M&A“):
   AcquisitionIndicator Company Company;
2. Отставка “Resignation“ (“Res“):
   ResignationIndicator Person;
3. Смена должности “ManagnentPositionChange“ (“MPC“):
   PositionChangeIndicator Person Position;

Поиск событий осуществляется в пределах одного предложения. Ситуация, когда событие покрывает сразу несколько предложений, не рассматривается. Ещё одним важным допущением является безоговорочное доверие механизму извлечения именованных сущностей[30]. То есть в процессе извлечения событий считаем именованной сущностью всё то, что таковым считает используемый Named Entity Recognizer. Если в качестве атрибута выступает именованная сущность и она определена неправильно, то, возможно, и событие определено неверно. В этом случае мы получим негативную оценку для правил, которые на самом деле не виновны в этой ошибке. Именно поэтому для оценки извлечения событий мы примем вышеописанное условие.

2 Обзор существующих методов

В данном разделе представлены некоторые алгоритмы извлечения информации, построения правил и обобщения регулярных выражений, статьи и исследования на которое опирается данная работа.

2.1 Извлечения событий

Задача извлечения событий поставлена в работе [14] с опорой на ACE (The Automatic Content Extraction) - программу для разработки механизмов автоматического извлечении информации[5]. Здесь же описаны некоторые возможные типы события.[6] В статье [24] обсуждаются возможности выбора наилучшего корпуса для обучения. Общий обзор методов извлечения информации представлен в [11]. Способы поиска новых типов события описаны в [22]. Работа [18] так же рассматривает события типа “M&A“. Здесь же утверждается, что большинство правил для извлечния такого рода событий создаются человеком. Авторы работы [32] извлекают события из новостных лент на китайском языке. Они применяют предметную онтологию, словари имён и в качестве главной структуры события используют связь SVO (Subject-Verb-Object).

2.2 Алгоритмы

Существует несколько алгоритмов построения правил:

2.2.1 Snowball и DEPREE

Алгоритм DEPREE [12, 11, 27] был предложен в 1998 году. DEPREE умеет извлекать бинарные отношения из интернет страниц используя разметку и URL.

Snowball [8, 11, 27] стал его логическим продолжением и также служит для поиска бинарных отношений в тексте, но не нуждается в разметке. Оба эти алгоритма используют небольшое множество положительных примеров. Текст обрабатывается итеративно: сначала положительные примеры отыскиваются в тексте, затем анализируется контекст найденных примеров; после в найденном контексте отыскиваются новые примеры, после чего процесс повторяется. Рассматриваемые алгоритмы применялись для поиска пар “Книга - Автор“ и “Организация - Местоположение“

К сожалению, не все события могут быть описаны с помощью бинарных отношений. Для бизнес-событий нельзя гарантировать постоянство, то есть в рассматриваемых примерах мы точно знаем, что автор уже написанной книги никогда не измениться и также местоположение компании, обычно сохраняется в течении длительного промежутка времени,

[6] http://www.ldc.upenn.edu/Projects/ACE/

но компания выступающая в роли покупателя сегодня, может завтра стать покупаемой, а человек в течение года может сменить несколько должностных позиций.

Ещё одним недостатком указанных алгоритмов можно назвать итеративную обработку корпуса документов. Корпус может оказаться достаточно большим и его обработка будет дорогостоящей.

2.2.2 ExDisco

Ещё один алгоритм, требующий примеры - ExDisco [33]. Используя эти примеры, он разделяет весь корпус на множество релевантных и нерелевантных документов и использует следующую концепцию: множество релевантных документов содержит релевантные события, и релевантные события содержатся в релевантных документах. После разделения каждое предложение рассматривается как кандидат на роль правила. Наиболее успешные кандидаты становятся новыми правилами. После этого обновляется классы релевантных и не релевантных документов.

Этот алгоритм также итеративно обрабатывает корпус. Кроме того, необходима заранее знать некоторую информацию об рассматриваемых документах для деления по релевантности, что может потребовать дополнительных усилий.

2.2.3 Wisk

Существует алгоритм, работающий с несколькими атрибутами, это Wisk [29]. Он использует не аннотированный корпус документов и синтаксический разбор предложения. Основным недостатком алгоритма Wisk является участие пользователя непосредственно в процессе обучения.

2.2.4 KnowItAll и TextRunner

Также существуют алгоритмы Open Event Extraction [15, 17], извлекающие информацию из множества не аннотированных Веб-документов. Наиболее известные из них - KnowItAll[15, 11, 16] and TextRunner[34, 11]. Эти алгоритмы используют поисковые запросы и не требуют заранее определённых отношений.

2.2.5 Извлечение сложных отношений

В работе [28] рассматривается алгоритм для извлечения не бинарных отношений. Он основан только на использовании частей речи. Он так же ищет события внутри одного предложения, но игнорирует отношения, выраженные не глаголом.

2.3 Построение правил

Помимо выше перечисленных источников, правила, в том числе линейные, для извлечения событий используют в работах [1, 18, 10] Использование регулярных выражений для обучения на аннотированном корпусе обсуждается в работе [19]. Авторы демонстрируют возможность применения их метода для поиска имён, url-ссылок и других сущностей.

Идея использования конвейера (англ. pipeline) не нова. В работе [9] так же рассматривается конвейер.

Для построения и обобщения правил в данной работе использовался подход, описанный в работе [23]. Основная идея состоит в том, что правила строятся на основе размеченных примеров. Самый первый вариант, в сущности, совпадает с текстом самого примера. Далее правило итеративно обобщается с помощью набора операций, и на каждую следующую итерацию переходит модификация правила, получившая самую высокую оценку.

Однако в данной работе используется другой критерий остановки алгоритма. Авторы используют два множества в процессе построения правила: и тестовое, и тренировочное. Мы используем только одно тренировочное множество для получения оценки правила. Как говорилось выше, линейные правила схожи по смыслу с регулярными выражениями. В той же работе [23] авторы рассматривают основные операции, которые можно проводить над правилами такого рода.

3 Алгоритм

Для работы механизма построения правил необходимы корпус неразмеченных документов и описание типов событий в виде xml-документа. На рисунке 1 приведен пример описания типа события “M&A“. В качестве атрибутов события здесь выступают именованные сущности типа *Company*. *Acquirer* и *acquiree* - роли атрибутов. В нашей работе роли рассматриваться не будут, для нас имеет значение только наличие атрибутов и их тип.

```
<relation-description relation-type="AcquisitionEvent" governor-type="AcquisitionIndicator" >
    <argument-types>
        <argument-type value="Company" key="acquirer"/>
        <argument-type value="Company" key="acquiree"/>
    </argument-types>
</relation-description>
```

Рис. 1: Описание типа события “M&A“.

Предлагаемый процесс построения правил состоит из четырёх этапов. В результате первого, на основе описания событий и синтаксического анализа, выделяются потенциальные события. На следующем этапе все события группируются по типу синтаксической связи между индикатором и атрибутами. Третьим этапом следует проверка пользователем примеров из каждой группы. В результате получается размеченный корпус и основа для построения правил. На последнем этапе строятся и обобщаются линейные правила в виде регулярных выражений.

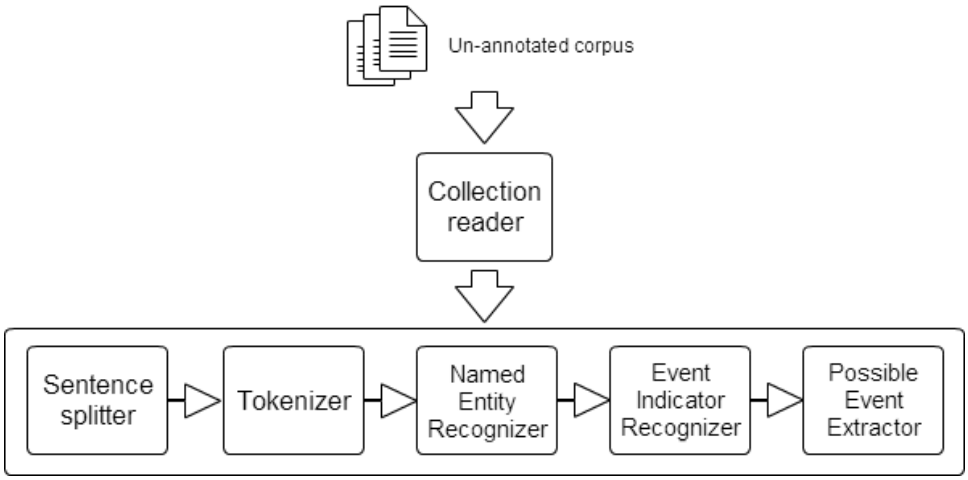

Рис. 2: Предобработка текста.

До начала работы алгоритма исходный текст подвергается предобработке(рис. 2). Каждый документ из корпуса делится на предложения. Каждое предложение делится на токены (слова). После этого в предложении при помощи OpenNLP библиотеки выявляется именованные сущности (Named Entity Recognition), такие, как *Company*, *Person* и другие.

Далее извлекаются индикаторы возможных событий с помощью словарей (обычно глагол или глагольная группа слов). Полученной информации достаточно, чтобы перейти к поиску потенциальных событий.

Как уже упоминалось, в данной работе подразумевается, что событие содержится исключительно в одном предложении. Если предложение содержит индикатор, то целесообразно продолжить поиск потенциального события в нем. Если же индикатор события не обнаружен, то рассматриваемое предложение пропускается.

3.1 Разметка корпуса

3.1.1 Синтаксический анализ текста

Каждое событие характеризуется индикатором и списком обязательных и дополнительных аргументов и атрибутов. Это можно записать следующим образом:

$$E\left[T, I, \{A_i^N, A_i^T\}_{i=1}^m\right], \tag{1}$$

где E - потенциальное событие, T - тип события, I - тип индикатора, событие имеет m аргументов и i-ый аргумент называется A_i^N и имеет тип A_i^T. Например, событие *Merger&Acquisition* описывается индикатором типа *AcquisitionIndicator* и двумя обязательными аргументами типа *Company*. Если в предложении есть индикатор и присутствуют атрибуты в достаточном количестве, то строятся все возможные события: для каждого индикатора рассматриваются все возможные сущности, подходящие на роль аргументов. В итоге получается набор всех возможных комбинаций из индикатора и атрибутов.

Можно точно оценить количество потенциальных событий N, которые генерируются для каждого индикатора. Предположим, что количество разных типов аргументов равно $m/$ и для каждого j-ого типа S_j является количеством аргументов в событии указанного типа. Если N_j - количество разных именованных сущностей рассматриваемого типа в предложении, то

$$C(N_j, S_j) = N_j!/(S_j!(N_j - S_j)!) \tag{2}$$

$$N = \prod_{j=1}^{m/} C(N_j, S_j) \tag{3}$$

Заметим, что 2 является формулой для сочетаний. Это является следствием того факта, что в нашей работе не учитывается информация о ролях атрибутов.

После формирования потенциального события вычисляется путь от каждого аргумента до индикатора в дереве синтаксических зависимостей. В результате для каждого события существует правило, содержа-

щее индикатор, атрибуты и синтаксические связи. Рассматривается самый короткий путь от множества слов индикатора до множества слов атрибута. Например: *AcquisitionIndicator Company:agent Company: nsubjpass TemporalExpression:agent/prep in.*

Вернёмся к рассматриваемому ранее примеру:

"Trade debt is about $10 million, a court paper said." Разбор предложения, приведённого выше с помощью синтаксического анализатора, будет выглядеть следующим образом:

nn(debt-2, Trade-1)
nsubj($-5, debt-2)
cop($-5, is-3)
quantmod($-5, about-4)
root(ROOT-0, $-5)
number($-5, 10-6)
number($-5, million-7)
det(paper-11, a-9)
nn(paper-11, court-10)
appos($-5, paper-11)
partmod(paper-11, said-12)

Рис. 3: Синтаксический разбор предложения *"Trade debt is about $10 million, a court paper said."* при помощи Stanford parser

Из рис. 3 видно что глагол "said", который выступает в роли индикатора связан с компанией "Trade" следующим путём: -partmod/-appos/nsubj/nn. Рассмотрим предложение с настоящим событием: *"Google has announced plans to acquire photo-sharing service Panoramio for an undisclosed amount".*

nsubj(announced-3, Google-1)
aux(announced-3, has-2)
root(ROOT-0, announced-3)
dobj(announced-3, plans-4)
aux(acquire-6, to-5)
xcomp(announced-3, acquire-6)
amod(Panoramio-9, photo-sharing-7)
nn(Panoramio-9, service-8)
dobj(acquire-6, Panoramio-9)
det(amount-13, an-11)
amod(amount-13, undisclosed-12)
prep_for(Panoramio-9, amount-13)

Рис. 4: Синтаксический разбор предложения *"Google has announced plans to acquire photo-sharing service Panoramio for an undisclosed amount."* при помощи Stanford parser

Здесь компания и индикатор связаны всего лишь одной простой связью: nsubj (рис. 4), которая очевидно указывает, что компания является подлежащим, а индикатор, выраженный глаголом, – сказуемым. Видно, что данное предложение действительно содержит в себе событие. Для более подробного понимания рисунков 3 и 4 можно обратиться к работе [13].

Формирование синтаксического пути даёт возможность отсеивать маловероятные результаты (если связь между атрибутом и индикатором превосходит заранее определённую длину).

Большим недостатком использования синтаксического анализатора для извлечения событий является его время исполнения[7]. Поэтому использование данного подхода для извлечения каждого отдельного события слишком дорого. Кроме того, использование синтаксического анализатора для непосредственного извлечения событий подразумевает под собой использование информации о том, какие синтаксические конструкции соответствуют событию, а какие - нет. Путём рассмотрения нескольких примеров мы можем выбрать несколько очевидно верных и очевидно ложных связей между атрибутами и индикатором для каждого типа событий. Но это скажет нам, во-первых, что они верны для данных конкретных предложений и нет гарантии, что они сработают на всех остальных; во-вторых, даже если мы рассмотрим очень много предложений, мы так же не сможем гарантировать, что больше не существует никаких сложных синтаксических конструкций, которые тоже могут описывать событие.

Далее мы будем работать в большей степени именно с потенциальными событиями. Однако каждому из них соответствует предложение, поэтому иногда мы будем говорить о предложении, подразумевая под ним потенциальное событие.

3.1.2 Группирование событий

Следующим шагом обработки корпуса является группирование предложений по типам синтаксических связей. Сформированные на предыдущем шаге синтаксические связи являются критерием группирования.

В данном разделе допускается следующее предположение: синтаксическая связь индикатора и аргументов однозначно определяет истинность рассматриваемого события. Описанная гипотеза находит своё обоснование в [13] и в [26]. Основная идея доказательства допущенного утверждения состоит в следующем: синтаксическая связь между двумя словами однозначно определяет смысл, который эта связь несёт для человека. Например, если мы имеем некоторое существительное связанное с глаголом синтаксической связью nsubj, то значит некоторый объект,

[7]были проведены эксперименты, в которых правила, сожержащие синтаксическую информацию уступили по времени исполнения линейным правилам, при этом приимуществ по точности, полноте и F1-мере не наблюдалось.

определенный рассматриваемым существительным, совершил некоторое действие, определяемое глаголом, и другого смысла здесь нет и быть не может. Другими словами данная связь уже описывает некоторое событие, тип которого можно определить непосредственно по принадлежности существительного и/или глагола к заранее определённым словарным словам или именованным сущностям. В данной работе для каждого типа события используется словарь глаголов или отглагольных существительных в разных формах. Например, в словарь события “M&A“ входят такие слова как **to aquered** (приобретать), **to bay** (покупать) и другие.

Далее из каждой группы выбирается несколько предложений для предоставления их пользователю. Они нужны для принятия более точного решения, поскольку одно единственное предложение, характеризующее целую группу, может иметь сложный синтаксис или тяжело восприниматься пользователем. Также в этом случае можно избежать неверного решения из-за ошибки разметки именованных сущностей или ошибки токенизаци.

3.1.3 Проверка пользователем

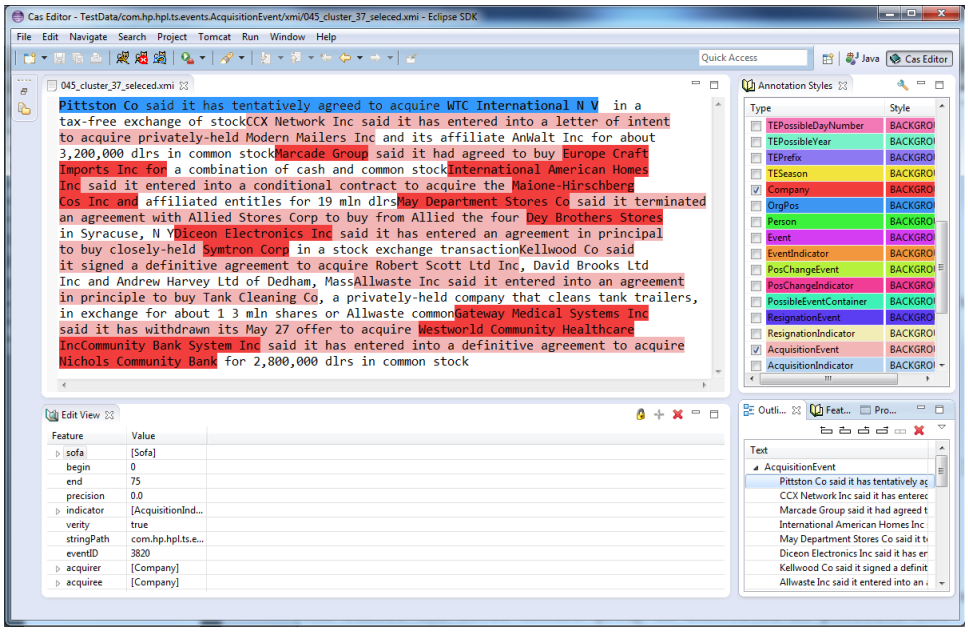

Рис. 5: CAS редактор. Инструмент UIMA Editor для Eclipse IDE.

Все потенциальные события сохраняются в особом CAS-формате. Пользователю предоставляется несколько предложений, которые были случайно выбраны из каждой группы. С помощью специального редактора CAS Editor для Eclipse IDE (рис. 5) предложения предстают для пользователя в графическом виде, что значительно упрощает проверку. Данный инструмент позволяет подсветить разными цветами в предложении

само потенциальное событие, а также его индикатор и атрибуты. В результате пользователю достаточно только ответить на вопрос - является ли рассматривая комбинация индикатора и атрибутов событием.

Первыми на проверку отправляются группы, содержащие наибольшее количества элементов. Предполагается, что именно они содержат наиболее часто встречаемые события. Группы, содержащие небольшое количество элементов, могут даже не рассматриваться. Например, можно не рассматривать синтаксические связи встретившиеся единожды и соответствующие им группы, содержащие всего один элемент. Даже если в этом случае будет упущено событие, оно мало повлияет на оценку работы всей системы.

После разметки пользователем каждой группе присваивается метка “истинности“ или “ложности“: если хотя бы половина размеченных пользователем предложений действительно содержит событие, вся группа отмечается как “истинная“; в противном случае она помечается как “ложная“. На основе групп, содержащих истинные события, строится размеченный корпус событий, который впоследствии используется как золотой стандарт. Часть этого корпуса становится основой для линейных правил, другая часть используется для оценки полученных правил.

3.2 Построение линейных правил

Завершающей стадией алгоритма является построение линейных правил. В качестве обучающего множества выступает часть золотого стандарта, полученного при помощи пользователя. Начальная версия правила представляет собой часть предложения в самом общем виде. Далее с помощью одной из четырёх операций оно приводится к более общему виду. К модифицированному правилу снова применяется одна из операций. Процесс продолжается до тех пор, пока улучшается F1-мера. На каждом шаге для следующей модификации выбирается правило, получившее более высокое значение F1-меры, чем остальные.

3.2.1 Структура линейных правил

Линейные правила для TextMARKER состоят из двух частей. В результате рассматриваемого алгоритма строится первая (левая) часть правила, которая представляется из себя нечто похожее на регулярное выражение. Она однозначно задаёт вторую, правую часть правила, которая содержит действие, т.е. пометить найденный с помощью первого шаблона выражение как событие. Более того правая часть правила содержит информацию о соответствии узлов правила и токенов в предложении. Далее рассмотрим левую часть правила и в последующих разделах под словом правило будем иметь ввиду только его левую часть. Правило состоит из узлов, которые имеют квантификатор, флаг обобщения и могут быть четырёх типов:

3.2.2 Флаг обобщения

1. Флаг обобщения указывает на возможность изменения узла правила одной из операций. Если он равен *true* узел можно модифицировать, если *false*, значит нельзя изменять тип или квантификатор рассматриваемого узла.

3.2.3 Квантификатор

Квантификатор состоит из двух частей:

1. Числовая часть - указывает на количество элементов рассматриваемого узла, которые могут следовать друг за другом. MIN значение $= 0$, MAX значение $= 25$.
   Пример: [7] - ровно семь элементов; $[0, 5]$ - от нуля до пяти элементов включительно.
   По умолчанию все узлы имеют числовую часть квантификатора равную [1] и она не отображается ни каким символом.

2. Строковая часть - особые дополнительные символы: “?“, “⋆“, “⋆?“ для квантификатора.

3.2.4 Типы узлов

1. **Узел аргумента (Argument Node).** Данный узел описывает индикаторы и атрибуты события. Они содержат только имя:
   AnnouncementIndicator, *AcquisitionIndicator*, *Person*, *Company*, и т.д.
   Флаг изменяемости рассматриваемого типа всегда равен *false*, т.е. не может быть изменён. Числовая часть квантификатора для таких узлов всегда равна [1] и строковая часть всегда отсутствует, потому что такой узел должен существовать и всегда в единственном экземпляре.

2. **Текстовый узел (Text Node).** Данный узел рассматривает конкретное слово или символ. Он содержит только строковый текст слов. По умолчанию такой узел может быть изменяем и строковая часть квантификатора отсутствует.

3. **Типовой узел(Type Node).** Данный узел описывает слова и символы как элементы системы типов TextMARKER(рис. 6).

4. **Узел регулярного выражения (RegExp Node).** Данный узел используется в особых случаях, когда нужно определить несколько возможных вариантов слова в правиле с помощью регулярного выражения.

3.2.5 Операции обобщения

1. *BottomUpGen*: Изменение (обобщение) типа узла (слова) – переход от частных слов к узлам из системы типов. Данный оператор состоит из двух уровней: первый уровень (level A) преобразует текстовые узлы (Text Nodes) в наиболее частные типовые узлы (Type Nodes). Второй уровень (level B) трансформирует типовые узлы в другие, более общие, типовые узлы системы типов TextMARKER[8].

   (a) Level A. Замена тестового узла на типовой узел:
   “recently“ → SW; “May“ → CW; “INC“ → CAP

   (b) Level B.

      i. замена типового узла другим, более общим, типовым узлом:
      SW, CW, CAP → W

      ii. замена пунктуационных знаков более общим типом:
      COLON, COMMA, SEMICOLON → PM

      iii. замена типовых узлов более общими типами:
      W, PM → ANY

   После того как тип узла изменён, его соседние узлы проверяются и если их тип совпадают, то они объединяются в один узел с более общим квантификатором:
   “company“ “based“ “in“ “Tel“ “Aviv“ → SW SW SW CW CW → SW[3] CW[2]
   на следующем шаге SW[3] CW[2] → W[5]

2. *QuantModif*: Изменение квантификатора. Расширение или сужение числовой части квантификатора.

   (a) Расширение: SW[3] → SW[2, 4]

   (b) Сужение: CW[2, 7] → CW[2, 6]

3. *AddCommaConstr*: Добавление конструкции с запятыми – замена содержимого, находящегося между двумя запятыми на любую последовательность слов. Любое количество символов (узлов) может быть заменены на один узел (ANY) если все заменяемые узлы можно изменять. После применения данной операции все узлы, покрываемые рассматриваемой конструкцией, становятся неизменяемыми и квантификатор становятся равным [1].

[8]SW - слово с маленькой буквы; CW - слово с большой буквы; CAP - слово, написанное только большими буквами; W - слово(более общий тип); COLOM - точка; COMMA - запятая; SEMICOLOM - точка с запятой; PM - знак припинания(внутри предложения), ANY - любой символ. Вся система типов представлена на рис. 6

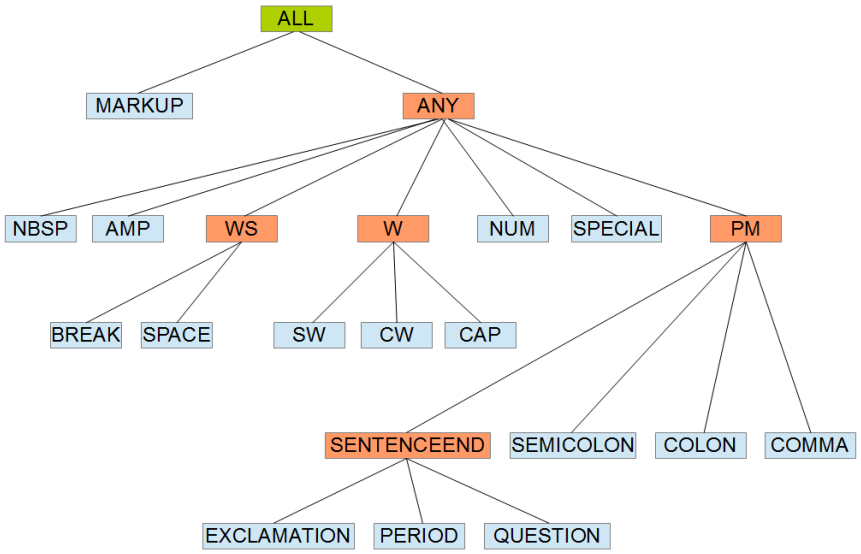

Рис. 6: Дерево типов системы TextMARKER.

(a) Company “,“ “Palo“ “Alto“ “base“ “company“ “,“ “today“ AnnouncementIndicator → Company COMMA ANY[1, 25]?* COMMA “today“ AnnouncementIndicator

(b) В следующем примере рассматриваемая операция не будет применена: Company “,“ “which“ AcquisitionIndicator Company “,“ → Company COMMA?* SW?* AcquisitionIndicator Company

4. *AddArticle*: Особая обработка артиклей в предложении: добавление регулярного выражения. Артикль в английском языке зависит от первой буквы следующего за ним слова и/или от присутствия последующего слова ранее в тексте, но это не существенно с точки зрения линейных правил. Однако наличия артикля в определённом месте предложения может быть существенно. Данная операция трансформирует текстовый узел в узел регулярного выражения(RegExp Node)

   (a) The University of Alabama has acquired an IBM Blue Gene/L supercomputer → CW{REGEXP(“The“, “A“, “An“)} Company AcquisitionIndicator SW{REGEXP(“the“, “a“, “an“)} Company

В процессе обобщения алгоритм запоминает все промежуточные варианты правила и их оценку, и если в последующем мы получаем один из уже рассмотренных ранее вариантов, обобщение для данного правила прекращается, поскольку результат уже известен. Так как оценка производится на одном и том же наборе данных, то нет необходимости на-

чинать построение правила с тех событий (предложений), которые уже были извлечены ранее построенными правилами.

4 Особенности реализации алгоритма

4.1 Генерация потенциальных событий

В процессе генерации потенциальных событий нужно учитывать, что встречается типы событий, у которых аргументами являются сущности одного и того же типа. Например, событие типа “M&A“ имеет в качестве аргументов две сущности тапа *Company*. В таком случае один и тот же атрибут не может занимать несколько слотов. Кроме того, так как мы в данной работе не рассматриваем роли атрибутов, если уже есть событие с атрибутом A, занимающим первый слот, и с атрибутом B занимающим второй слот, нет необходимости генерировать события, в которых атрибут B займёт первый слот, а атрибут A второй, т.е. тоже самое событие, где атрибуты поменяны местами. Так же не представляет интереса событие, в котором на месте атрибутов находятся разные сущности, имеющие один и тот же текст. Например, если в предложении встречается несколько раз название одной и той же компании, то каждому вхождению соответствует отдельная сущность, но текст идентичен, как и значение для человека.

Обойти упоминание одной и той же компании в качестве разных атрибутов достаточно просто - необходимо просто проверить на несоответствие текст, который покрывают рассматриваемые сущности. Для того, что бы строились события без дубликатов, используется описанный ниже механизм. Все рассматриваемые сущности внутри предложения упорядочены по положении в предложения. На каждой новой итерации к потенциальному событию добавляются только те сущности, индекс которых больше или равен номеру итерации на которой он добавляется. Такой подход позволяет избежать построения потенциальных событий, отличающихся только порядком атрибутов.

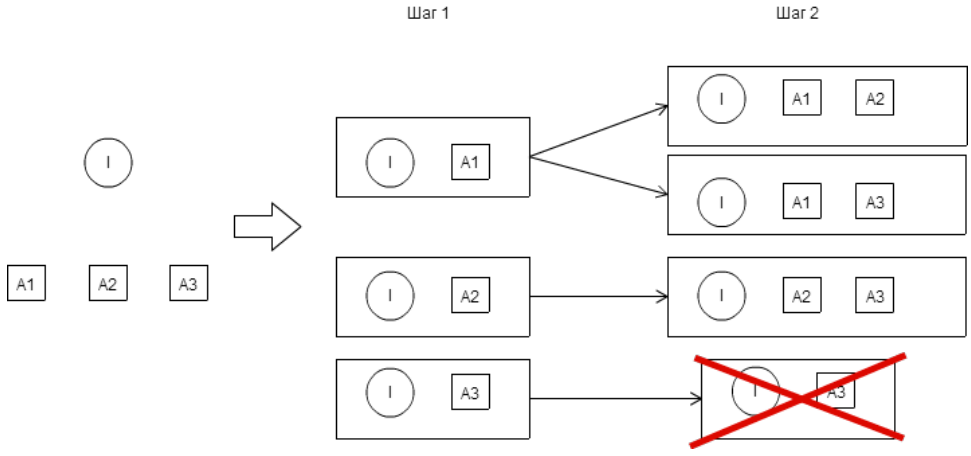

Рис. 7: Схема построения потенциальных событий

На рисунке 7 изображен пример такого построения. Допустим в неко-

тором предложении есть индикатор некоторого события (I), которому нужны два атрибута и три сущности, претендующие на роль этих атрибутов (A1, A2, A3). На первом шаге появятся три возможных аннотации потенциального события. На втором шаге первое возможное потенциальное событие пополнится второй и третьей сущностью; второе пополнится только третьей сущностью. Третье возможное событие не сможет пополниться и удалится. В результате получится три потенциальных события, представляющие собой все возможные комбинации сущностей.

4.2 Кэширование F1-меры

В процессе обобщения правил на каждом шаге построения получается целое семейство правил, из которых необходимо выбрать наилучшее для следующей итерации. Оценка правила требует его применения и это очень затратная операция. При этом один и тот же вариант правила может возникать в разных ситуациях. Поэтому, при первом появлении правила считается его F1-мера и сохраняется. Далее уже не происходит непосредственного применения правила, мы пользуем уже вычисленное и сохранённое значение.

4.3 Индикатор события

Алгоритм обучается на тренировочном множестве - набор примеров в процессе обучения не меняется. Следовательно, если в результате обобщения было получено правило, извлекающее некоторый набор примеров, то не имеет смысла обучаться на правилах из этого набора. Потому что если для некоторого события мы получили правило, извлекающее его, так как корпус не меняется, то и оценка относительно этого корпуса не изменится. Значит, ничего лучшего получить нельзя. Если бы можно было получить более качественно правило, оно было бы получено при первом поиске.

Принимая во внимания описанный факт, система запоминает индикатор тех событий, для которых уже найдено извлекающее правило и пропускает их при дальнейшим обучении. Описанная идея находит своё обоснование в книге [25].

Отметим, что результат обучения может зависеть от точки входа. Другими словами, в зависимости от порядка поступления примеров для обучения может меняться результат. В рамках данной работы не рассматривался поиск оптимальной последовательности обучающих примеров (предложений). Для получения стабильного результата все события, которые выступают в качестве обучающих примеров, упорядочиваются по возрастанию идентификатора.

5 Анализ результатов экспериментов

5.1 Оценка качества

Что бы понять насколько хорошо или плохо построенные правила, извлекающие события, необходим некоторый механизм оценки. Работа [20] посвящена особенностям оценки извлечения информации и рассматривает различные подходы. В нашей работе в качестве метрик рассматриваются точность(P), полнота (R) и F1-мера:

$$P = \frac{tp}{tp + fp} \tag{4}$$

$$R = \frac{tp}{tp + fn} \tag{5}$$

$$F1 = \frac{2PR}{P + R} \tag{6}$$

где tp - верно найденное событие; fp - неверно найденное событие (найдено и отмечено как событие то, что таковым не является); fn - неверно не найденное событие(верное событие, которое не было найдено). Формулы 5-7 описываются во многих источниках: [11, 22, 8, 34, 29].

5.2 Описание данных

В данной работе были проведены две серии экспериментов над двумя разными корпусами документов, содержащие новостные статьи на английском языке.

Сравнивались результаты (точность, полнота и F1-мера) извлечения событий правилами, полученных в результате работы описанного процесса построения правил, и результаты извлечения событий с помощью правил, которые были написаны вручную человеком. Правила, построенные в ручную, основаны на анализе RSS статей, полученных из таких источников как новостные ленты Yahoo[9] и Google[10]. Список используемых правил можно найти в приложении A.

Первая серия экспериментов проводилась над данными Reutuers[11]. Использовался многоязыковой корпус RCV2. Из него были выбраны статьи на английском языке. Вторая серия экспериментов проводилась над другими данными.

В процессе разметки в спорных ситуациях отдавалось предпочтение “ложной“ метки. Как уже упоминалось, события, помеченные как истинные, далее составляют золотой стандарт, на части которого будет обучаться алгоритм. Обучаясь на относительно некорректных данных, ал-

[9] http://news.yahoo.com/
[10] https://news.google.com/
[11] http://www.reuters.com/

горитм построит правило, которое также будет извлекать некорректные или ложные события. Однако, если по ошибке, верное событие оказалось отмеченным как ложное, это ещё не означает, что не будет построено правило, размечающее его. Вообще говоря, группы предложений, поделённые по синтаксической связи и группы событий, извлекаемые одним и тем же правилами не эквивалентны. В оной синтаксической группе могут лежать предложения, извлекаемые разными правилами, и одно линейное правило может так же извлекать события из нескольких синтаксических групп.

5.3 Первая серия экспериментов

Как уже упоминалось выше, рассматриваемый корпус Reutuers (RCV2). При построение групп игнорировались группы, содержащие всего один элемент. Такие группы считаем не существенными для обработки всего текста. Остальные группы были размечены вышеописанной способом. В итоге получился корпус из 83, 45, 80 предложений, из которых 12, 16, 25 содержат истинные события для типов события “M&A“, “MPC“ и “Res“ соответственно (таблица 1). Корпуса для каждого события были

Таблица 1: Описание корпуса данных

Event type	M&A	MPC	Res
Sentence #	83	45	80
True events #	12	16	25

случайно разбиты на тренировочное и тестовое множество примерно в соотношении 2:1. Так как данных получилось не очень много, для каждого события повторили эксперимент с другой разбивкой. В таблице 2 представлены результаты разбиения на тестовое и тренировочное множество для каждого из рассматриваемых типов событий.

Таблица 2: Разделение на тренировочное и тестовое множество

	Train sentence #	Test sentence #
M&A(1)	61	22
M&A(2)	55	28
MPC(1)	38	7
MPC(2)	30	15
Res(1)	54	26
Res(2)	56	24

Результаты по каждому эксперименту и каждому типу события представлены в таблицах 3 - 5. Рисунки 8 - 13 графически отображают содержание таблиц 3 - 5.

Таблица 3: Результат для правил, построенных человеком (MC) и построенных автоматически (AC) для события *Mergers & Acquisitions*(“M&A“)

M&A (1)	MC Train	MC Test	AC Train	AC Test
Rules #	10	-	8	-
Precision	0.5	-	1.0	1.0
Recall	0.11	0.0	1.0	0.33
F1-measure	0.18	-	1.0	0.5
M&A (2)				
Rules #	10	-	8	-
Precision	-	0.5	1.0	1.0
Recall	0.0	0.5	1.0	0.5
F1-measure	-	0.5	1.0	0.66

Для события “M&A“ на тренировочном и тестовом множествах оценка извлечения событий правилами, построенными автоматически, превзошли по всем параметрам (точность, полнота и F1-мера) оценку извлечения событий, построенных вручную человеком, в обоих экспериментах. Отметим, что система построила только 8 правил, а человеком были построено 10. Также заметим, что в первом эксперименте на тестовом множестве и во втором эксперименте на тренировочном множестве правила построенные человеком не смогли извлечь вообще ни одного события.

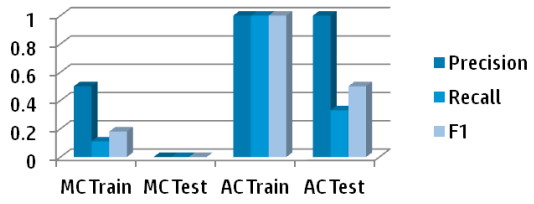

Рис. 8: Значение точности, полноты и F1-меры для эксперимента 1 события M&A

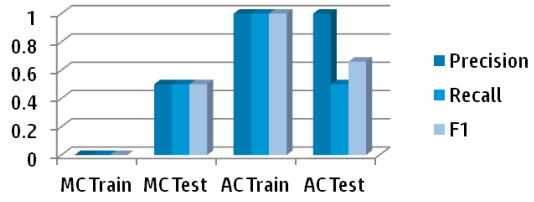

Рис. 9: Значение точности, полноты и F1-меры для эксперимента 2 события M&A

Для события “MPC“ правила, построенные автоматически, также отработали не хуже по всем метрикам, чем правила написанные человеком.

Таблица 4: Результат для правил, построенных человеком (MC) и построенных автоматически (AC) для события *Management Position Change*(“MPC“)

MPC (1)	MC Train	MC Test	AC Train	AC Test
Rules #	3	-	9	-
Precision	0.78	0.5	0.92	1.0
Recall	0.58	0.33	1.0	0.33
F1-measure	0.67	0.4	0.96	0.5
MPC (2)				
Rules #	3	-	8	-
Precision	0.67	1.0	1.0	1.0
Recall	0.6	0.4	1.0	0.4
F1-measure	0.63	0.57	1.0	0.57

Но здесь система построила 9 правил, человеком были построены только три.

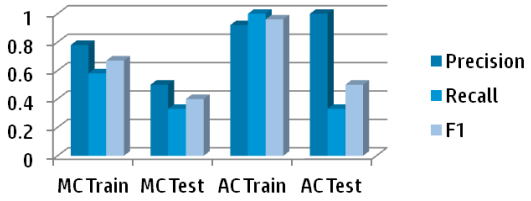

Рис. 10: Значение точности, полноты и F1-меры для эксперимента 1 события MPC

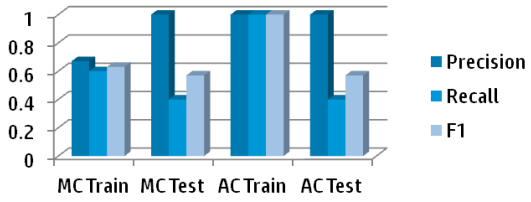

Рис. 11: Значение точности, полноты и F1-меры для эксперимента 2 события MPC

В экспериментах проведённых для события “Res“ правила, написанные человеком в двух тестах уступили правилам, написанным человеком по точности, но извлечённые события имели крайне низкую полноту, поэтому по измерению F1-меры правила, построенные рассматриваемой системой, оказались лучше в обоих экспериментах и на тренировочном и на тестовом множествах. Для указанного типа событий автоматически построенных правил получилось в два раза больше, чем построенных вручную: 10 и 5 соответственно.

Таблица 5: Результат для правил, построенных человеком (MC) и построенных автоматически (AC) для события *Resignation* (Res)

Res (1)	MC Train	MC Test	AC Train	AC Test
Rules #	5	-	10	-
Precision	1.0	1.0	0.9	0.33
Recall	0.19	0.25	0.9	0.75
F1-measure	0.32	0.4	0.9	0.46
Res (2)				
Rules #	5	-	10	-
Precision	1.0	1.0	0.83	0.5
Recall	0.18	0.33	0.86	0.67
F1-measure	0.3	0.5	0.84	0.57

Таблица 6: Результат измерения F1-меры при применении автоматически построенных правил в перекрёстных экспериментах, проведённых пять раз

Запуск	M&A	MPC	Res
1	0.4523	0.5428	0.5000
2	0.4666	0.5714	0.5988
3	0.4888	0.6750	0.6321
4	0.5555	0.6904	0.6426
5	0.8333	0.7023	0.6764
Среднее	0.5593	0.63638	0.60998
Отклонение	0.1582	0.0737	0.0674

Так же были проведены перекрёстные эксперименты для пяти множеств. В ходе этого эксперимента измерялось только значение F1-меры. Так как данных относительно не много, эксперимент повторили пять раз. В таблице 6 и на рисунке 14 представлены результат значения F1-меры упорядоченные по возрастанию. Как можно видеть значение F1-меры для события “M&A“ изменяются от 0,45 до 0,83 (почти в два раза). Для события MPC от 0,54 до 0,7 и для Res от 0,5 до 0,68. Это свидетельствует о том, что данных не достаточно для проведения подобного рода экспериментов и результат получается статистически не обоснованным.

Ко всему имеющемуся корпусу документов были применены правила, построенные человеком. Результаты представлены в таблице 7 и рисунке 15. Отметим что для событий “M&A“ и “Res“ значение F1-меры оказалось ниже чем нижнее значение F1-меры в перекрёстных экспериментов для правил, построенных автоматически. Для события MPC правила построенные автоматически превзошли правила, построенные вручную человеком, в трёх запусках из пяти.

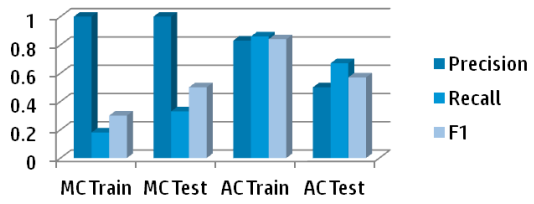

Рис. 12: Значение точности, полноты и F1-меры для эксперимента 1 события Res

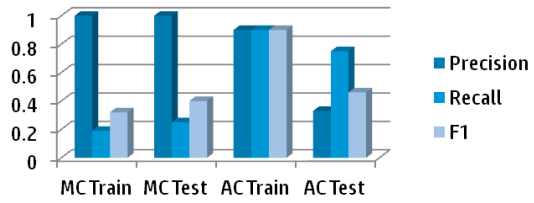

Рис. 13: Значение точности, полноты и F1-меры для эксперимента 2 события Res

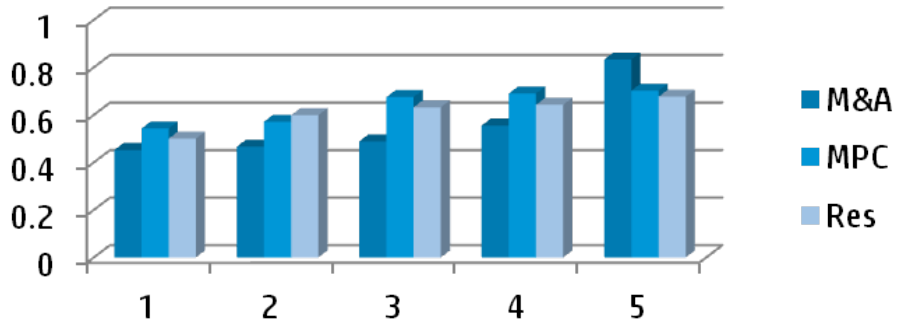

Рис. 14: Значение F1-меры для перекрёстных экспериментов

5.4 Вторая серия экспериментов

После некоторых дополнений и оптимизаций была проведена ещё одна серия экспериментов на более обширных данных, но рассматривалось всего два вида события “M&A“ и “MPC“. В таблице 8 представлен размер корпуса и разбиение на тренировочное и тестовое множество: для события “M&A“ было выбрано 1379 предложений, из которых 264 содержали события (тренировочное множество состояло из 924 предложений, тестовое из 455). Для события MPC всего было рассмотрено 1511 предложений, из которых 1070 содержали события (в тренировочное множество вошли 1028 предложений, остальные 483 составили тестовое).

Так же, как и в первой серии экспериментов, мы будем измерять точность, полноту и F1-меру. Помимо извлечения событий правилами, по-

Таблица 7: Результат применения правил, написанных вручную ко всему корпусу данных

	M&A	MPC	Res
Precision	0.5	0.73	1.0
Recall	0.08	0.53	0.2
F1-measure	0.14	0.62	0.33

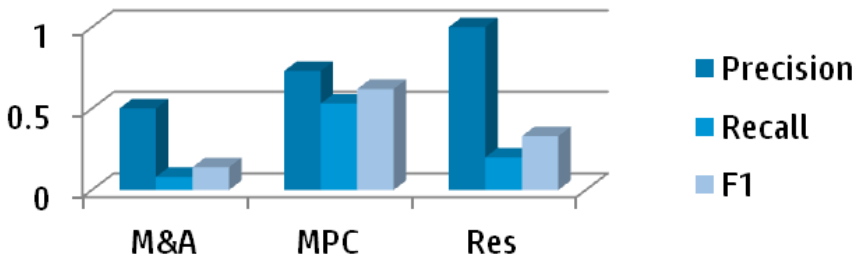

Рис. 15: Значение точности, полноты F1-меры правил, написанных человеком применённых ко всему корпусу

строенными в ручную(MCB), и правилами, построенными системой(AC), мы попытались извлечь события правилами, которые имеют самый общий вид в системе TextMARKER (MCC). Список самых общих правил приведён в приложении B. Отметим, что во второй серии экспериментов было наложено ещё одно ограничение на построенные правила: в финальное множество не включаются правила, которое в самом обобщенном своём варианте извлекают только одно событие, на котором оно строилось. При принятых ограничениях множество правил, которые строит система, имеют более читабельный вид. При этом значение F1-меры ухудшается незначительно [12].

В таблице 9 и на рисунке 16 представлен результат указанных метрик для события MPC. Правила, имеющие самый общий вид показали максимальную полноту, равную 1, но относительно низкую точность (0.35) и на тренировочном, и на тестовом множествах. В результате чего значение F1-меры также не велико - 0.52. Оставшиеся два набора

[12]Были проведены соответствующие эксперименты

Таблица 8: Описание корпуса

Event type	M&A	MPC
Sentence #	1379	1511
Positive events #	264	1070
Train set #	924	1028
Test set #	455	483

Таблица 9: Результат для правил, имеющих самый общий вид (MCC), правил, построенных человеком (MCB) и правил, построенных автоматически (AC) для события *"Management Position Change"* (MPC)

(MPC)	MCC Train	MCC Test	MCB Train	MCB Test	AC Train	AC Test
Rules #	2	-	3	-	9	-
Precision	0.36	0.35	0.85	0.87	0.85	0.69
Recall	1.0	1.0	0.67	0.68	0.67	0.95
F1-measure	0.52	0.52	0.75	0.77	0.74	0.8

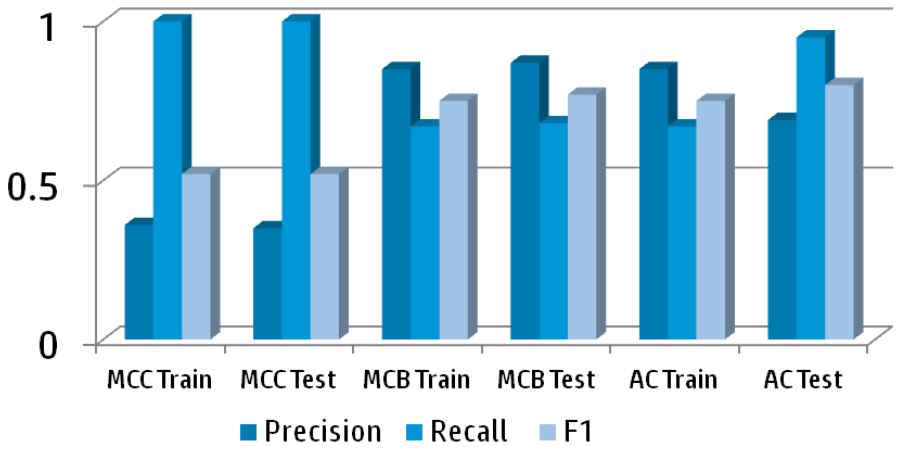

Рис. 16: Значение точности, полноты F1-меры правил, написанных человеком применённых ко всему корпусу

правил отработали абсолютно одинаково на тренировочном множестве - все метрики совпали. На тестовом множестве правила, построенные человеком, превзошли правила, построенные системой, по точности (0.87 против 0.69), но значительно уступили в полноте (0.68 против 0.95). В результате F1-мера правил, построенных человеком, оказалась выше, чем правила построенных человеком (0.75 против 0.8).

В таблице 10 и на рисунке 17 представлен результат указанных метрик для события "M&A". Правила, имеющие самый общий вид, также показали максимальную полноту, равную 1 и очень низкую точность (0.04) и на тренировочном, и на тестовом множествах. Значение F1-меры крайне мало - 0.07 и 0.08 на тренировочном и тестовом множествах. Правила, построенные системой, на тренировочном и тестовом множествах, показали более высокий результат по всем критериям, чем правила построенные человеком: точность 0.67 против 0.8 и 0.78; полнота 0.29 против 0.8 и 0.76; F1-мера 0.39 и 0.4 против 0.8 и 0.77 для тренировочного и тестового множества соответственно.

Таблица 10: Результат для правил, имеющих самый общий вид (MCC), правил, построенных человеком (MCB) и правил, построенных автоматически (AC) для события *“Mergers & Acquisitions“*(“M&A“)

(M&A)	MCC Train	MCC Test	MCB Train	MCB Test	AC Train	AC Test
Rules #	3	-	10	-	5	-
Precision	0.04	0.04	0.67	0.67	0.8	0.78
Recall	1.0	1.0	0.27	0.29	0.88	0.76
F1-measure	0.08	0.07	0.39	0.4	0.8	0.77

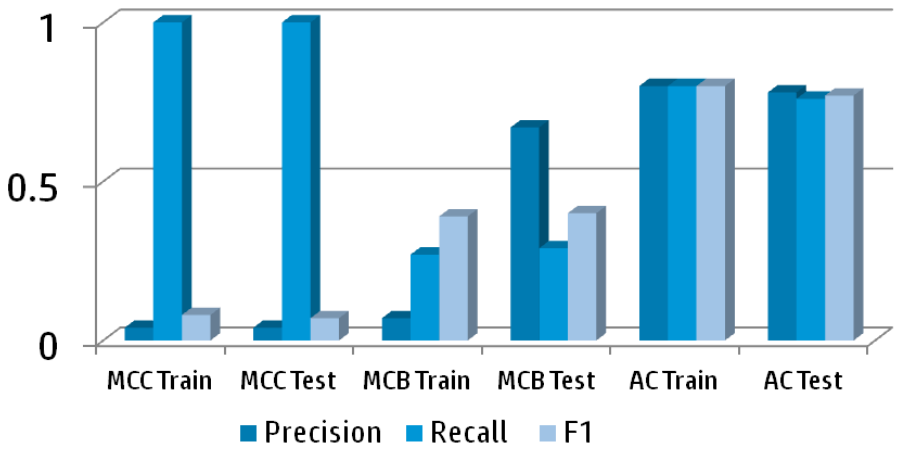

Рис. 17: Результат для правил, имеющих самый общий вид (MCC), правил, построенных человеком (MCB) и правил, построенных автоматически (AC) для события *“Mergers & Acquisitions“*(“M&A“)

Заключение

В представленной работе были рассмотрены и проанализированы механизмы извлечения событий и алгоритмы построения правил. С учётом анализа достоинств и недостатков существующих методов были предложены механизмы автоматизированной разметки корпуса и построения линейных правил. Реализована системы разметки корпуса и построения линейных правил для системы TextMARKER.

Реализация протестирована на синтетических данных. Кроме этого, было размечено два множества реальных новостных статей методом, описанным в нашей работе. Были проведены эксперименты на упомянутых множествах для типов события “M&A“, “Res“ и “MPC“. Полученные в результате работы алгоритма правила показали свои преимущества перед правилами, которые были построены вручную человеком, по измерению F1-меры во всех проведённых экспериментах. По точности и полноте правила построенные системой оказались лучше в большинстве экспериментов.

Список литературы

[1] А. Андреев, Д. Березкин, and К. Симаков. Модель извлечения фактов из естественно-языковых текстов и метод ее обучения. In *Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды восьмой всероссийской научной конференции (RCDL'2006)*, page 252–262. Ярославский государственный университет, 2006.

[2] В. Ярец. 50 самых распространенных языков в мире, 2009.

[3] В. Сараев. Когда данные стали большими, 2013.

[4] В. Постолатий. Bigdata шагает по планете, 2013.

[5] *ACE (Automatic Content Extraction) English Annotation Guidelines for Events*, 5.4.3 2005.07.01 edition, 2005.

[6] Usage of content languages for websites, 2011.

[7] Объем информации в мире растет с высокой скоростью, 2013.

[8] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 85–94, New York, NY, USA, 2000. ACM.

[9] D. Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia, July 2006. Association for Computational Linguistics.

[10] T. Arrskog, P. Exner, H. Jonsson, P. Norlander, and P. Nugues. Hyperlocal event extraction of future events. In *In Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2012)*, pages 11–20, Boston, 2012. CEUR Workshop Proceedings.

[11] N. Bach and S. Badaskar. A Review of Relation Extraction. 2007.

[12] S. Brin. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on The World Wide Web and Databases*, WebDB '98, pages 172–183, London, UK, UK, 1999. Springer-Verlag.

[13] M.-C. de Marnee and C. D. Manning. Stanford typed dependencies manual.

[14] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. *Proceedings of LREC 2004*, pages 837–840, 2004.

[15] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, Dec. 2008.

[16] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[17] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Center. Open Information Extraction: The Second Generation. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[18] F. Hogenboom, F. Frasincar, U. Kaymak, and F. de Jong. An overview of event extraction from text. Oct. 2011.

[19] P. Kluegl, M. Atzmueller, and F. Puppe. Integrating the rule-based ie component textmarker into uima. In J. Baumeister and M. Atzmueller, editors, *LWA-2008 (Special Track on Information Retrieval)*, pages 73–77, 2008.

[20] A. Lavelli, M. E. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. Ie evaluation: Criticisms and recommendations, 2004.

[21] Lewis, M. Paul, G. F. Simons, and C. D. F. (eds.). Summary by language size, 2013.

[22] H. Li, X. Li, H. Ji, and Y. Marton. Domain-independent novel event discovery and semi-automatic event annotation.

[23] Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. V. Jagadish. Regular expression learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 21–30, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[24] S. Liao and R. Grishman. Can document selection help semi-supervised learning?: a case study on event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 260–265, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[25] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[26] D. McClosky, M. Surdeanu, and C. D. Manning. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1626–1635, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[27] R. McDonald. Extracting relations from unstructured text. Technical report, University of Pennsylvania, 2005.

[28] R. McDonald, F. Pereira, S. Kulick, S. Winters, Y. Jin, and P. White. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 491–498, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[29] S. Soderland. Learning information extraction rules for semi-structured and free text. *Mach. Learn.*, 34(1-3):233–272, Feb. 1999.

[30] M. Tkachenko and A. Simanovsky. Named entity recognition: Exploring features. In J. Jancsary, editor, *Proceedings of KONVENS 2012*, pages 118–127. ÖGAI, September 2012. Main track: oral presentations.

[31] D. Vasserman. Самый распространенный язык в мире, 2012.

[32] S.-H. Wu, T.-H. Tsai, and W.-L. Hsu. Domain event extraction and representation with domain ontology. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 33–38, 2003.

[33] R. Yangarber, R. Grishman, and P. Tapanainen. Automatic acquisition of domain knowledge for information extraction. In *In Proceedings of the 18th International Conference on Computational Linguistics*, pages 940–946, 2000.

[34] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, and S. Soderland. Textrunner: Open information extraction on the web. In *HLT-NAACL (Demonstrations)'07*, pages 25–26, 2007.

Приложение

Приложение А: линейные правила, построенные человеком

Событие “Merge&Acquisition“:

1. Company AcquisitionIndicator Company->GATHER (AcquisitionEvent,1,2,3,“acquiree“=1,“indicator“=2,“acquirer“=3);
2. Company AcquisitionIndicator “by“ Company->GATHER (AcquisitionEvent,1,2,3,4,“acquiree“=4,“acquirer“=1,“indicator“=2);
3. Company “(“ AcquisitionIndicator “by“ Company“)“- >GATHER (AcquisitionEvent,1,2,3,4,5,6,“acquiree“=5,“acquirer“=1,“indicator“=3);
4. Company AcquisitionIndicator “the“? Company ->GATHER (AcquisitionEvent,1,2,3,4,“acquiree“=1,“acquirer“=4,“indicator“=2);
5. Company AcquisitionIndicator “the“? “privately“ “held“ Company ->GATHER (AcquisitionEvent,1,2,3,4,5,6,“acquiree“=1,“acquirer“=6,“indicator“=2);
6. Company AcquisitionIndicator “the“? “privately“ “-“ “held“ Company- >GATHER (AcquisitionEvent,1,2,3,4,5,6,7,“acquiree“=1,“acquirer“=7,“indicator“=2);
7. Company COMMA “a“ ANY*? COMMA AcquisitionIndicator “the“? Company->GATHER (AcquisitionEvent,1,2,3,4,5,6,7,8, “acquiree“=1,“acquirer“=8,“indicator“
8. Company SPECIAL “s“ AcquisitionIndicator “of“ Company ->GATHER (AcquisitionEvent,1,2,3,4,5,6, “acquiree“=1, “acquirer“=6,“indicator“=4);
9. Company SPECIAL “s“ AcquisitionIndicator Company->GATHER (AcquisitionEvent,1,2,3,4,5,“acquiree“=1,“acquirer“=5,“indicator“=4);
10. Company WREGEXP(“agreed|about“) AcquisitionIndicator “the“? Company- >GATHER (AcquisitionEvent,1,2,3,4,5,“acquiree“=1,“acquirer“=5,“indicator“=3);

Событие “Resignation“:

1. Person “accepted“ “the“? ResignationIndicator W*? Person ->GATHER(ResignationEvent,4,6,“person“=6,“indicator“=4);
2. Person ResignationIndicator ->GATHER(ResignationEvent,1,2,“person“=1,“indicator“=2);
3. Person COMMA ANY*? COMMA ResignationIndicator ->GATHER(Resignation,1,2,3,4,5,“person“=5);
4. Person “has“? “decided“ “to“? ResignationIndicator ->GATHER(Resignation,1,2,3,4,5,“person“=1,“indicator“=5);

Событие “ManagnentPositionChange“:

1. Person COMMA ANY*? COMMA PosChangeIndicator
   ->GATHER(PosChangeEvent,1,2,3,4,5,6,7,“person“=1,“indicator“=5);

2. Person PosChangeIndicator
   ->GATHER(PosChangeEvent,1,2,“person“=1,“indicator“=2);

3. PosChangeIndicator Person
   ->GATHER(PosChangeEvent,1,2,“person“=2,“indicator“=1);

Приложение B: линейные правила, имеющие самый общий вид

Событие “Merge&Acquisition“:

1. Company ANY*? AcquisitionIndicator ANY*? Company->GATHER
   (AcquisitionEvent,1,2,3,4,5,“acquiree“=1,“indicator“=3,“acquirer“=5);

2. AcquisitionIndicator ANY*? Company ANY*? Company->GATHER
   (AcquisitionEvent,1,2,3,4,5,“acquiree“=3,“indicator“=1,“acquirer“=5);

3. Company ANY*? Company ANY*? AcquisitionIndicator->GATHER
   (AcquisitionEvent,1,2,3,4,5,“acquiree“=1,“indicator“=5,“acquirer“=3);

Событие “ManagnentPositionChange“:

1. Person ANY*? PosChangeIndicator
   ->GATHER(PosChangeEvent,1,2,3,“person“=1,“indicator“=3);

2. PosChangeIndicator ANY*? Person
   ->GATHER(PosChangeEvent,1,2,3,“person“=3,“indicator“=1);