Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем
Информационные системы и базы данных

Бодрова Анастасия Александровна

Разрешение кореференции методом кластеризации

Магистерская диссертация

Научный руководитель:
к. ф.-м. н., доцент Графеева Н. Г.

Рецензент:
к. т. н., руководитель службы разработки
инструментов качества поиска Браславский П. И.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Department of Analytical Information Systems
Information Systems and Databases

Anastasiya Bodrova

Coreference resolution using clusterization

Master's Thesis

Scientific supervisor:
associate professor Natalia Grafeeva

Reviewer:
head of search quality tools development Pavel Braslavski

Saint-Petersburg
2016

Оглавление

1. Введение

Разрешение кореференции является одной из ключевых подзадач обработки текста [5]. Для точной работы систем глубокого понимания текста, например, направленных на извлечение информации (в т.ч. именованных сущностей), разрешение кореференции является необходимым этапом.

Разрешение кореференции является задачей объединения текстовых упоминаний, относящихся к одной сущности дискурса. Текстовые упоминания могут быть выражены именами собственными, именной группой (существительное с зависимыми от него словами) или местоимениями. В Таблице 1 приведён пример возможных текстовых упоминаний сущности "Джо Смит" [1].

Имя собственное:	Джо Смит, Мистер Смит
Именная группа:	парень в синей рубашке
Местоимение:	он, ему

Таблица 1: Примеры возможных упоминаний сущности "Джо Смит"

Интерес к задаче разрешения кореференции возрос при появлении отдельной дорожки в 1995 году на соревновании в рамках Конференции по Пониманию Сообщений, MUC-6 (Message Understanding Conference), организованном компанией NRAD при поддержке DAPRA.

К настоящему времени разработано множество систем по разрешению кореференции. Чаще всего, они используют подходы машинного обучения, и улучшают их с помощью различных лингвистических и концептуальных признаков. Ключевыми типами моделей являются попарная модель (mention-pair model), модель ранжирования (ranking model) и сущностная модель (entity-based model) [21]. Современные исследования направлены на методы машинного обучения "без учителя"[15] и сущностные модели [4].

2. Постановка задачи

В данной работе мы поставили себе целью примененить алгоритм кластеризации для разрешения кореференции на русскоязычных новостных текстах. Мы начали с более узкой задачи: кластеризации имён собственных, которые относятся к персонам (Person-type). Другими словами, задача заключается в объединении всех доступных в тексте частей имени (т.е. обращение, имя, фамилия, отчество) каждой персоны, упомянутой в тексте. Например, в романе "Война и мир" Л.Толстого, мы бы объединили упоминания "князь Андрей", "князь Андрей Николаевич" и "князь Болконский" в одну сущность "князь Андрей Николаевич Болконский".

Наш алгоритм включал в себя два этапа:

1. *Извлечение упоминаний*
   Для извлечения именованных сущностей были написаны грамматики для Томита-парсера. Томита-парсер - свободно доступный инструмент, использующий формализм контексто-свободных грамматик и газеттиры для извлечения фактов.

2. *Кластеризация*
   Для объединения извлечённых имён в сущности, мы использовали аггломеративную кластеризацию: изначально каждое упоминание находится в собственном одноэлементном кластере, затем на каждом шаге два кластера объединяются. Объединение кластеров происходит на уровне сущностей: любая пара упоминаний между объединяемыми кластерами не должна содержать противоречий.

Мы проводили эксперименты на новостных текстах, размеченных для соревнования Dialogue Evaluation factRuEval-2016 [32], а так же сравнили результаты с участниками соревнования.

3. Обзор литературы

Предыстория разрешения кореференции начинается с ближайшей задачи - разрешения анафоры. Разрешение анафоры - это задача нахождения предшествующего упоминания (называемое антецедентом), на которое ссылается местоимение. Например, в тексте "Фёдор вошёл в комнату. Он выглядел потерянным.", местоимение "он" относится к имени "Фёдор". Одно из главных отличий между анафорой и кореференцией заключается в том, что результатом разрешённой анафоры является множество пар "местоимение-его упоминание-антецендент", тогда как задача разрешения кореференции заключается в нахождении кластеров упоминаний, каждый из которых относится к отдельной сущности дискурса.

Вычислительные теории дискурса, в частности фокусирование(focusing) [10] и, позднее, *центрирование(centering)*[12] [11], имели значительное влияние на исследования в области разрешения кореференции в 1970-1980 гг., что привело к росту количества алгоритмов, основанных на центрировании [28].

Работа над разрешением кореференции начала активно развивается с появлением отдельной дорожки в 1995 году на соревновании в рамках Конференции по Пониманию Сообщений, MUC-6 (Message Understanding Conference), организованном компанией NRAD при поддержке DAPRA. В 1998, дорожка по разрешению кореференции была также включена в MUC-7. По итогам этих соревнований появились массивы размеченных данных, что стимулировало применение алгоритмов машинного обучения к задаче кореференции. С тех пор используются три основных класса моделей разрешения кореференции: *попарная модель (mention-pair model)*, *модель ранжирования (ranking model)* и *сущностная модель (entity-based model)*.

Попарная модель

Попарная модель является классификатором, который определяет, являются ли упоминания в паре кореферентными. Решения о корефе-

рентности каждой пары, принимаюся независимо друг от друга. Наиболее распространёнными алгоритмами кластеризации являются *closest-first*[26] [27] и *best-first*[22] [31] [2].

Сущностная модель

В отличие от попарной модели, сущностная модель классифицирует пары упоминаний, используя информацию из уже созданных (в т.ч. частично) кластеров. Данный подход позволяет получать больше информации для принятия решения, которой часто бывает недостаточно при попарном сравнеии. Более того, эта модель поддерживает транзитивность, не требуя постобработки.

Существует классический пример, показывающий преимущество сущностной модели перед попарной. Например, исходный текст содержит три упоминания: "Мистер Клинтон", "Клинтон" и "Хилари Клинтон". В попарной модели мы независимо соединяем "Мистер Клинтон" и "Клинтон", а так же "Хилари Клинтон" и "Клинтон", и тогда, исходя из транзитивности, получаем, что "Мистер Клинтон" и "Хилари Клинтон" принадлежат одному кластеру, что является неверным результатом из-за противоречий по роду. Однако при использовании сущностной модели, объединив в один кластер "Мистер Клинтон" и "Клинтон", мы не сможем связать упоминания "Хилари Клинтон" и "Клинтон" из-за несовпадений по роду, потому что значение рода у упоминания "Клинтон" появилось от значения рода на уровне сущности, которое, в свою очередь, появилось от упоминания "Мистер Клинтон".

Первая попытка применения сущностной модели была предпринята в работе Luo et al. [18], в которой все возможные кластеризации были представлены в виде дерева Белла, а задача разрешения кореференции рассматривалась как нахождение наилучшего пути от корня к листьям.

В других исследованиях предлагались различные стратегии оптимизации кластеризации: вероятностная модель первого порядка, подбирающая признаки на основе логики первого порядка[9]; целочисленное линейное программирования для поддержки транзитивности [6] [8] [7]; алгоритмы разделения графов [17] [23]; использование имитационного

обучения и комбинирование моделей [4].

Модель ранжирования

Модель ранжирования считается промежуточным шагом между попарной и сущностными моделями. При использовании данной модели на одном шаге может быть проверено более одного упоминания-кандидата, таким образом проводится соревнование между всеми кандидатами и выбирается наиболее подходящий.

В Ng (2005) [19] использовано другое применение ранжирования: задача кореференции в этой работе определена как ранжирование потенциальных кластеризаций, сгенерированных различными попарными моделями. Rahman and Ng (2009) [25] предложили алгоритм ранжирования кластеров, который объединяет преимущества модели ранжирования и сущностной модели.

Модель обучения без учителя

Интерес к использованию моделей обучения без учителя мотивирован отсутствием или ресурсоёмкостью размеченных данных.

Впервые данная модель была использована в Cardie and Wagstaff(1999) [3]. Их подход основывался на кластеризации векторов признаков, отражающих упоминания.

Исследователи продолжают изучать возможности данного типа модели. Haghighi and Klein (2007) [13] предложили попарную непараметрическую порождающую Байесовскую модель. Ng (2008) [20] усовершенствовал их модель с помощью EM-кластеризацию. Poon and Domingos (2008) [24] предложили сущностную модель с использованием Марковской логики. Ma and Hovy (2016) [16] применили порождающую модель ранжирования.

4. Модель

4.1. Извлечение упоминаний

В данной работе мы будем считать упоминанием имя собственное[30], являющиеся личным именем.

Личные имена могут состоять из имени, фамилии, отчества и прозвища. Мы использовали данную классификацию на основе стандартной структуре полного русского личного имени [29]. Данная структура подходит для большинства славянских языков, однако в других языках имена могут содержать дополнительные части, например, титул, второе имя или матчество. В нашей работе мы не опираемся на культурные особенности при обработке имён, а лишь разделяем их на имя и фамилию. Например, арабское имя *Мухаммад ибн Салман ибн Амеен аль-Фарси*, означающее *Мухаммад, сын Салмана, сын Амеена, из Персии*, будет обработано как имя - *Мухаммад* и фамилия - *ибн Салман ибн Амеен аль-Фарси*

4.1.1. Инструмент

Для извлечения упоминаний мы используем Томита-парсер[33], свободнодоступный иструмент для извлечения структурированных данных из текстов на естественном языке. Парсер основан на алгоритме GLR-парсера [14] и использует формализм контексто-свободных грамматик. Томита-парсер анализирует текст при помощи грамматик, описывающих правила, по которым далее выбираются цепочки слов, и использующих газеттиры (внутренние словари ключевых слов). В Таблице 2 проиллюстрирован пример правила и строки, которая эквивалентна ему.

4.1.2. Обработка текста

Целью этапа извлечения упоминаний является получение множества упоминаний с признаками из исходного текста. Для достижения этой цели, мы написали две грамматики для Томита-парсера. Первая

Person ->	kwtype<address>	nob_part	Word<surname>
	Графиня	де	Монсеро

Таблица 2: Пример правила для Томита-парсера. *Person* - нетерминал для полного личного имени. *kwtype<address>* - любое слово из словаря ключевых слов, состоящий из обращений. *nob_part* - нетерминал для фамильных приставок. *Word<surmame>* - терминал с семантическим признаком "фамилия"

.

из них используется на этапе предобработки, с её помощью мы извлекаем несловарные имена и заполняем ими временные словари. Вторая грамматика, используя созданные словари, собирает части имени вместе и извлекает соответствующие признаки. Обе грамматики используют созданные вручную словари ключевых слов, описывающие стоп-слова, дескрипторы географических объектов и организаций, обращения и другое. Модель извлечения упоминаний изображена на Рис.1

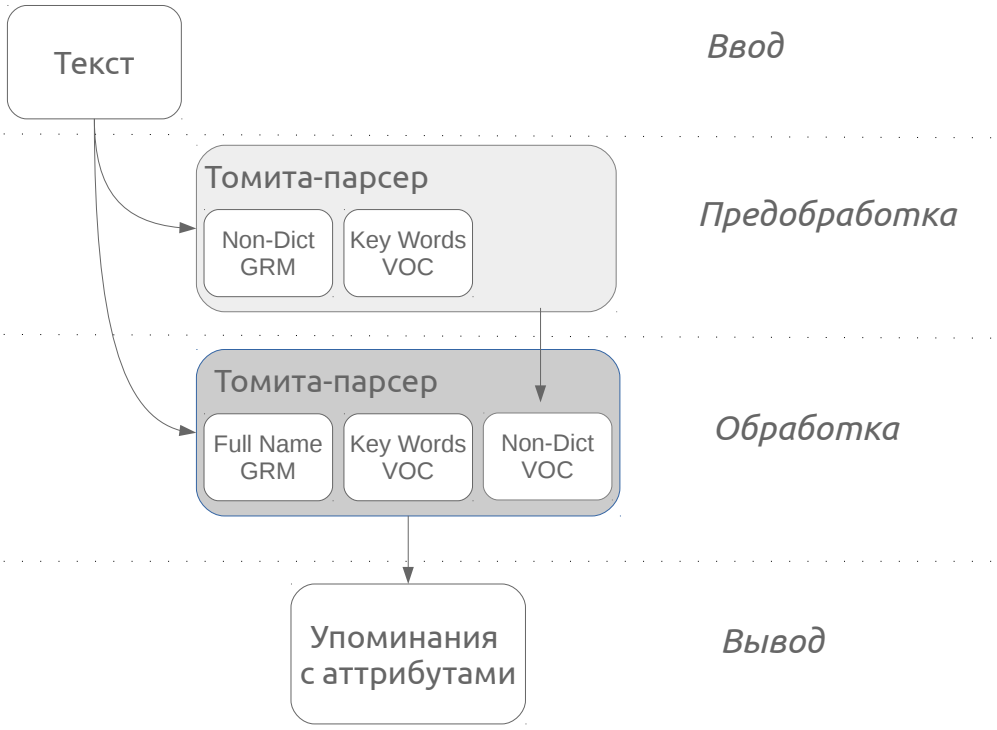

Рис. 1: Модель извлечения упоминаний. GRM - грамматики, VOC - словари ключевых слов

Грамматика для извлечения несловарных имён Томита-парсер использует внутренний словарь, который включает в себя личные имена. Однако практически невозможно внести в словарь все имена, которые могут появиться в тексте, но можно описать их с помощью грамматических и синтаксических правил.

Основная проблема данного этапа заключается в отделении имён, которые относятся к персонам, от имён, относящихся к другим типам сущностей (например, локаций или организаций). Для этой цели, мы создаём правила, которые описывают контекст, который с высокой вероятностью будет относиться к определённому типу сущности. Например, слова из словарей ключевых слов, которые включают в себя дескрипторы географических объектов и организаций (Приложение А)., реже бывают контактны с именами персон, чем с названиями локаций

или организаций.

Если несловарное имя появляется в определённом контексте, оно получает метку "pers_check", для контекста, свойственного персоне, и "misc_check", для всех остальных. Те имена, которые помечены как имена персон, добавляются во временно созданный словарь, другие добавляются в стоп-лист. В результате, мы получаем словарь ключевых слов(имён), который будет подключён к следующей грамматике.

Грамматика для извлечения несловарных имён представлена в Приложении B.

Грамматика для извлечения полного имени После создания словаря несловарных имён, мы можем извлекать полные имена, сформированные из их составляющих: Имя (Firstname), Фамилия(Lastname), Отчество(Patronymic), и несловарное имя, которое может быть любой из вышеупомянутых частей. Также, на данном этапе мы извлекаем некоторые дополнительные аттрибуты, такие как Род(Gender), Обращение(Address) и Дескриптор(Descriptor), которые описаны ниже.

При извлечении все имена нормализуются с помощью встроенного анализатора Томита-парсера. Нормализация - это процесс приведения слова к его нормальной форме (к именительному падежу, единственному числу), например, *"Борису"* -> *"Борис"*.

В результате мы получаем множество упоминаний с признаками. Грамматика для извлечения полного имени представлена в Приложении C.

Признаки упоминаний

Каждое упоминание представлено набором признаков, описанных ниже. Распределение признаков представлено на Рис. 4.1.2.

1. **NameNotNorm**

   Значением признака является строка упоминания в неизменном виде

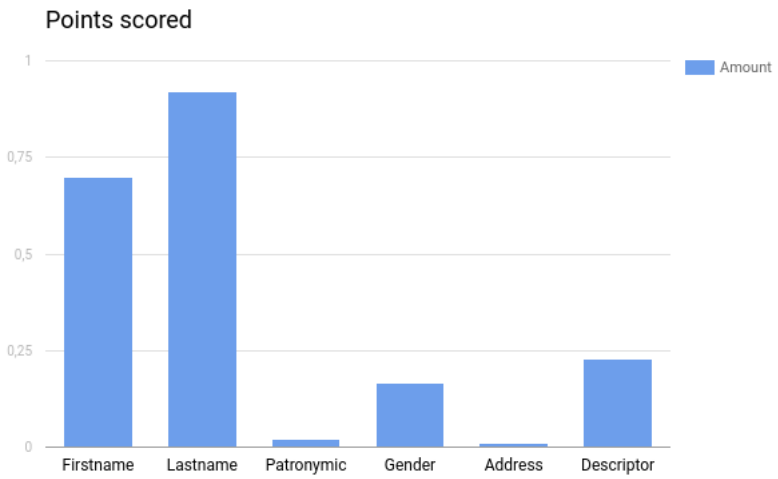

Рис. 2: Распределение извлечённых признаков для упоминаний

2. **Firstname**

3. **Lastname**

4. **Patronymic**

   Признаки Firstname, Lastname и Patronymic заполняются в соответствии с внутренним словарём Томиты-парсера. Несловарные слова мы соотносим с признаками на основе их лексических и синтаксических особенностей. Например, если извлечённое имя заканчивается на *"швили"* - то с большой вероятностью, это слово является грузинской фамилией, или, если слово стоит перед отчеством, то скорее всего является именем.

5. **Gender**

   Признак Gender может принимать любое из трёх значений: *"male"*, *"female"*, *"dual"*. Род определяется несколькими способами. Во-первых, мы используем словарную информацию, если она доступна. Во-вторых, мы используем род обращений, описанных в словаре ключевых слов (Приложение А). Затем, мы смотрим на грамматический род синтаксически связанных глаголов, прилагательных, дескрипторов. Если имеющейся информации недостаточно для определения рода, то признак примет значение *"dual"*.

6. **Address**

   Значение признака Address может принимать любое слово из словаря обращений (Приложение А). Обращение является именной группой, которая используется как дополнительная часть имени, при разговоре с человеком (например, *"мистер"*, *"лорд"*)

7. **Descriptor**

   Дескриптор является именной группой, описывающий некоторый класс, к которому принадежит человек (например, профессия, или родственный статус).

   Более 80% дескрипторов в новостных текстах появляются с первым упоминанием персоны. Это связано с необходимостью предоставить больше информации о впервые упоминаемом герое. Следовательно, с большей вероятностью мы встретим имя с дескриптором до имени той же персоны без него.

Ниже представлен процесс извлечения упоминаний из одного предложения:

Текст:
В США содержатся в заключении без обвинений два журналиста: фотограф Ассошиэйтед Пресс Билал Хассеин (Bilal Hussein), и оператор Аль-Джазиры Сами аль-Хадж (Sami al-Haj), которые находится в тюрьме уже пять лет и в настоящий момент содержится в Гуантанамо (Куба).

Вывод грамматики для извлечения несловарных имён:
В США содержатся в заключении без обвинений два журналиста: фотограф ***[Ассошиэйтед]*** *Пресс Билал* ***[Хассеин]*** *(Bilal Hussein), и оператор Аль-Джазиры* ***[Сами] [аль-Хадж]*** *(Sami al-Haj), которые находится в тюрьме уже пять лет и в настоящий момент содержится в Гуантанамо (Куба).*

RawName
{
Name = Ассошиэйтед
NameNotNorm = Ассошиэйтед
Misc_check = true
}
RawName
{
Name = Хассеин
NameNotNorm = Хассеин
Pers_check = true
}
RawName
{
Name = Сами
NameNotNorm = Сами
Pers_check = true
}
RawName
{
Name = аль-Хадж
NameNotNorm = аль-Хадж
Pers_check = true
}

Сгенерированные словари:

Имена: хассеин, сами, аль-хадж

Другое: ассошиэйтид

Вывод грамматики для извлечения полного имени:

В США содержатся в заключении без обвинений два журналиста:

фотограф Ассошиэйтед Пресс ***[Билал Хассеин]*** *(Bilal Hussein), и оператор Аль-Джазиры* ***[Сами аль-Хадж]*** *(Sami al-Haj), которые находится в тюрьме уже пять лет и в настоящий момент содержится в Гуантанамо (Куба).*

StrictName
{
    NameNotNorm = Билал Хассеин
    Firstname = Билал
    Lastname = Хассеин
    Patronymic = None
    Gender = male
    Address = None
    Descriptor = фотограф Ассошиэйтед Пресс
}
StrictName
{
    NameNotNorm = Сами аль-Хадж
    Firstname = Сами
    Lastname = аль-Хадж
    Patronymic = None
    Gender = male
    Address = None
    Descriptor = оператор Аль-Джазиры
}

Упоминания с признаками:
1: ('Билал', 'Хассеин', 'None', 'male', 'None', 'фотограф Ассошиэйтед Пресс')
2: ('Сами', 'аль-Хадж', 'None', 'male', 'None', 'оператор Аль-Джазиры')

4.2. Кластеризация

4.2.1. Определения

Пусть каждый документ $\mathcal{D}$ представлен в виде множества из n упоминаний $\mathcal{M} = \{m_1, ...m_n\}$. Все упоминания формируют множество уникальных пар упоминаний $\mathcal{MP} = \{(i,j) \mid 1 \leq i < j \leq n\}$, где каждое упоминание представлено собственным индексом.

Каждая пара упоминаний из $\mathcal{MP}$ имеет свой вес $\mathcal{P}_{(i,j)\in\mathcal{MP}}$, являющийся результатом функции $score(i,j)$, отражающей насколько два упоминания кореференты:

$$score(i,j) = \begin{cases} None, & if Disjoint(i,j) \\ \theta^T f(i,j), & otherwise \end{cases}$$

где $f(i,j)$ - вектор признаков для двух упоминаний, а θ -вектор весов признаков, полученный результате обучения на размеченных данных (4.2.2). Функция *Disjoint* показывает, существуют ли между двумя упоминаниями противоречия, из-за которых они не могут относиться к одной сущности.

Для постановки задачи разрешения кореференции, будем считать $n \times n$ Булеву треугольную матрицу C результатом *кластеризации*, где $C_{ij} = 1$, если m_i и m_j кореференты, иначе - 0. Кластеризация C является *валидной* тогда и только тогда, когда поддеживается ограничения по транзитивности: $(C_{ij} = 1 \wedge C_{ij} = 1) \Rightarrow (C_{ik}) = 1 \ \forall \ 1 \leq i < j < k \leq n$. Пусть $C[i]$ означает кластер, которому принадлежит i-ое упомиание.

Целью задачи разрешения кореференции является *валидная кластеризация* C на документе $\mathcal{D}$. Этап кластеризации описан в 4.2.4.

4.2.2. Модель пар упоминаний

Мы натренировали нашу модель пар упоминаний с помощью логистического классификатора, чтобы получить веса признаков. Модель предсказывает, принадлежат ли два упоминания к одному кластеру. Вероятность кореференции принимает стандартную логистическую фор-

му:

$$p_\theta(i,j) = (1 + e^{-\theta^T f(i,j)})^{-1}$$

где $f(i,j)$ - вектор признаков для m_i и m_j, а θ вектор весов признаков, который мы хотим обучить.

Мы считаем, что $\mathcal{M}$ является множеством всех упоминаний в обучающем множестве, пусть $\mathcal{T}(j)$ - множество индексов упоминаний, предшествующих j, таких, что m_i и m_j кореферентны, и $\mathcal{F}(j)$ - такие, что m_i и m_j не кореферентны. Множества $\mathcal{T}(j)$ и $\mathcal{F}(j)$ формируются для каждого документа $\mathcal{D}$, которому принадлежит упоминание m_j. Мы хотим найти вектор весов признаков θ, присваивающий высокие вероятности кандидатам из $\mathcal{T}(j)$ и низкие - кандидатам из $\mathcal{F}(j)$.

Модель натренирована на нашем обучающем множестве с помощью решения следующей задачи оптимизации с использованием L1-регуляризации:

$$\mathcal{L}(\theta) = -\sum_{m \in \mathcal{M}} \left(\sum_{t \in \mathcal{T}(m)} log\left(p_\theta\left(t,m\right)\right) + \sum_{f \in \mathcal{F}(m)} \left(log\left(1 - p_\theta\left(f,m\right)\right)\right) \right) + min\left\|\theta\right\|_1$$

Вектор признаков

Вектор признаков отражает совпадения между различными значениями признаков двух упоминаний.

Признаки **Firstname**, **Lastname** и **Patronymic** имеют 5 вариантов совпадения:

- *Strict Match (строгое совпадение)*
  Значения признака идентичны.

- *Relaxed Match (неполное совпадение)*
  Значения признака практически идентичны.

  Для проверки на неполное совпадение, мы рассчитываем расстояние Левенштейна между двумя строковыми представлениями признаков. Расстояние Левенштейна представляет собой строковую метрику для нахождения минимального числа однобуквенных пре-

образований (вставка, замена, удаление), необходимых для превращения одного слова в другое.

Проверка на неполное совпадение необходима для нахождения одинаковых имён с поправкой на ошибки нормализации и текстовые опечатки. Мы считаем строки близкими друг другу, если расстояние Левенштейна между ними не превышает 3 - максимальной длины окончания существительного в русском языке.

Также значение признака Firstname Match равно Relaxed Match, если одна строка является частью другой: *'Анна Мария'* и *'Анна'*.

- *First Letter Match (совпадение по первой букве)* Если значение хотя бы одного из признаков является сокращенным до одной буквы именем, и эта буква совпадает с первой буквой другого значения, то Firstname Match = First Letter Match.

- *None Field*
  Если хотя бы одно из полей принимает значения *'None'*.

- *Mismatch*
  Значения признаков полностью отличаются.

Признаки **Gender** и **Address** имеют только 3 варианта совпадений: Match, Mismatch и None Field. Признаки Gender и Address могут принимать значения ограниченного числа слов, таким образом, не сталкиваются с ошибками, опечатками или сокращениями.

Признак **Descriptor** отражает только наличие дескриптора у каждого из упоминаний.

4.2.3. Парный вес

Как написано выше, все упоминания в документе формируют множество уникальных пар, и каждая из них имеет свой собственный вес, который отражает, насколько близки упоминания, чтобы оказаться в одном кластере, т.е. быть кореферентыми.

id	Признак	Значение	Пример
1	Firstname_match	1. Strict Match 2. Relaxed Match 3. First Letter Match 4. None Field 5. Mismatch	”Андрей” ”Андрей” ”Андрею” ”Андрей” ”Андрей” ”А.” ”Андрей” ”None” ”Андрей” ”Мария”
2	Lastname_match	1. Strict Match 2. Relaxed Match 3. First Letter Match 4. None Field 5. Mismatch	”Иванов” ”Иванов” ”Иванову” ”Иванов” ”Иванов” ”И.” ”Иванов” ”None” ”Иванов” ”Петров”
3	Patr_match	1. Strict Match 2. Relaxed Match 3. First Letter Match 4. None Field 5. Mismatch	”Иванович” ”Иванович” ”Иванович” ”Ивановичу” ”Иванович” ”И.” ”Иванович” ”None” ”Иванович” ”Петрович”
4	Gender_match	1. Match 2. None Field 3. Mismatch	”female” ”female” ”female” ”None” ”female” ”male”
5	Addr_match	1. Match 2. None Field 3. Mismatch	”мистер” ”мистер” ”мистер” ”None” ”князь” ”король”
5	Descr_presence	1. Both true 2. First is true 3. Second is true 4. Both false	”true” ”true” ”true” ”false” ”true” ”true” ”false” ”false”

Таблица 3: Признаки для пар упоминаний

Функция парного веса может вернуть значение *'None'*, если у упоминаний есть противоречия и они определённо не кореферентны, иначе функция возвращает сумму взвешенных признаков для пар упоминаний.

$$score(i,j) = \begin{cases} None, & if\ Disjoint(i,j) = true \\ \theta^T f(i,j), & otherwise \end{cases}$$

Функция *Disjoint* имеет следующий вид:

$$Disjoint(i,j) = \begin{cases} true, & if\ Firstname\_match[Mismatch] = true \\ & or\ Lastname\_match[Mismatch] = true \\ & or\ Gender\_match[Mismatch] = true \\ & or\ \theta^T f(i,j) < 0 \\ false, & otherwise \end{cases}$$

В результате мы получаем множество $\mathcal{P} = \{score(i,j) \mid (i,j) \in \mathcal{MP}\}$.

4.2.4. Алгоритм кластеризации

Кластеризация происходит с использованием весов пар упоминаний и ограничением по транзитивности.

Широко распространённый подход кластеризации для разрешения кореференции является *best-first кластеризация* [22]. Для каждого упоминания, алгоритм выбирает наиболее подходящее предшествующее упоминание, которому оно может быть кореферентно.

Слабость данного подхода заключается в принятии решении на локальном, парном уровне, а не на уровне сущности. Таким образом, мы можем получить кластер с конфликтующими элементами, как в приведённом раннее примере,[*"Хилари Клинтон", "Клинтон", "Билл Клинтон"*]. Это происходит потому, что решение о кореферентности в паре "Хилари Клинтон" и "Клинтон" принимается отдельно от решение в паре "Билл Клинтон" и "Клинтон".

Чтобы избежать эту проблему, мы вводим уровень сущности на этапе объединения кластеров, проверяя, не принимает ли функция парных весов хотя бы одной из пар любых упоминаний из двух кластеров значение *'None'*, говорящем о том, что в кластерах есть строго некореферентные элементы.

Процесс объединения кластеров происходит с помощью аггломеративной кластеризации: каждое упоминание начинается в своём собственном одноэлементном кластере, и затем, на каждом шаге, два кла-

стера объединяются.

Во-первых, мы сортируем все пары упоминаний в убывающем порядке, основываясь на их парных весах. Это позволяет принимать сперва легкие решения, а затем, увеличив количество доступной информации, перейти к более сложным.

Затем, мы итерационно идём по сортированному списку. Для каждой пары мы принимаем решение о том, нужно ли объединить кластеры, в которых содержатся элементы. Решение зависит от наличия противоречий между любыми двумя упоминаниями из обоих кластеров. Алгоритм 1 описывают процедуры кластеризации.

Algorithm 1 Аггломеративная кластеризация

Input: Множество парных весов $\mathcal{P}$
Output: Кластеризация $\mathcal{C}$

1: $\mathcal{S} \leftarrow Sort_{desc}(\mathcal{P}_{(i,j)} \mid score(i,j) \neq None)$
2: **for** $(i,j) \in \mathcal{S}$ **do**
3: **if** $\mathcal{C}[i] \neq \mathcal{C}[j]$ **then**
4: **if** $CanMerge(\mathcal{C}[i], \mathcal{C}[j])$ **then** $Merge(\mathcal{C}[i], \mathcal{C}[j])$
5:

5. Эксперимент

Мы проводили эксперименты на материалах соревнования factRuEval-2016[32], организованного коммандой Dialogue Evaluation. Данные представляют собой 254 размеченных текста, 122 из которых были даны в виде обучающего множества(development set), и 132 были тестовым множеством(test set) и использовались для оценки (Таблица 4). Для того, чтобы сравнить наши результаты с участниками соревнования, мы тренировали нашу модель только на обучающем множестве.

	Development Set	Test Set
#Документов	122	132
#Упоминаний	741	1388
#Сущностей	391	643
Среднее # упоминаний	6.07	10.5
Среднее # сущностей	3.22	4.87

Таблица 4: Статистика по обучающему и тестовому множествам

В рамках соревнования были представлены три дорожки: Извлечение имён собственных, Разрешение кореференции, Извлечение фактов. Таким образом, мы можем отдельно сравнить наши результаты по первым двум дорожкам.

Также мы можем сравнить наши результаты с встоенными в Томита-парсер алгоритмами извлечения имён и их кластеризации.

Результаты алгоритмов оценивались с помощью традиционных метрик оценки: Precision (точноть), Recall(полнота) и F-measure (F-мера). Точностью является отношение найденных примеров к релевантным, а Полнота - релевантых к найденным. F-мера является гармоничной мерой между Точностью(P) и Полнотой(R): $2 \cdot \frac{P \cdot R}{P+R}$.

Все результаты отсортированы по F-мере.

5.1. Результаты этапа извлечения упоминаний

Извлечение упоминаний было первой дорожкой на соревновании factRuEval, в котором приняли участие 13 комманд (в таблице они обозначены названиями ц).

Томита-парсер предоставляет встроенный агоритм для извлечения личных имён. Он находит имена из внутреннего словаря в тексте, а затем комбинирует контактные имена, не имеющие противоречий по роду, числу и падежу. Мы использовали результаты Томита-парсера в качестве базового уровня.

Результаты сравнения на тестовом множестве представлены в Таблице 5 и Рис.3.

	Precision	Recall	F-measure
violet	0.9450	0.9155	0.9300
crimson	0.9620	0.8829	0.9208
black	0.9114	0.9236	0.9175
pink	0.9636	0.8677	0.9132
aquamarine	0.9080	0.9174	0.9127
grey	0.9556	0.8726	0.9122
наш подход	0.8964	0.9131	0.9047
beige	0.9274	0.8785	0.9023
brown	0.9608	0.8395	0.8961
purple	0.8972	0.8805	0.8888
green	0.9300	0.8403	0.8829
orange	0.9486	0.7861	0.8597
алгоритм Томита-парсера для извлечения имён	0.9428	0.7862	0.8574
white	0.9537	0.7399	0.8333
ruby	0.9169	0.7270	0.8110

Таблица 5: Результаты извлечения упоминаний на тестовом множестве

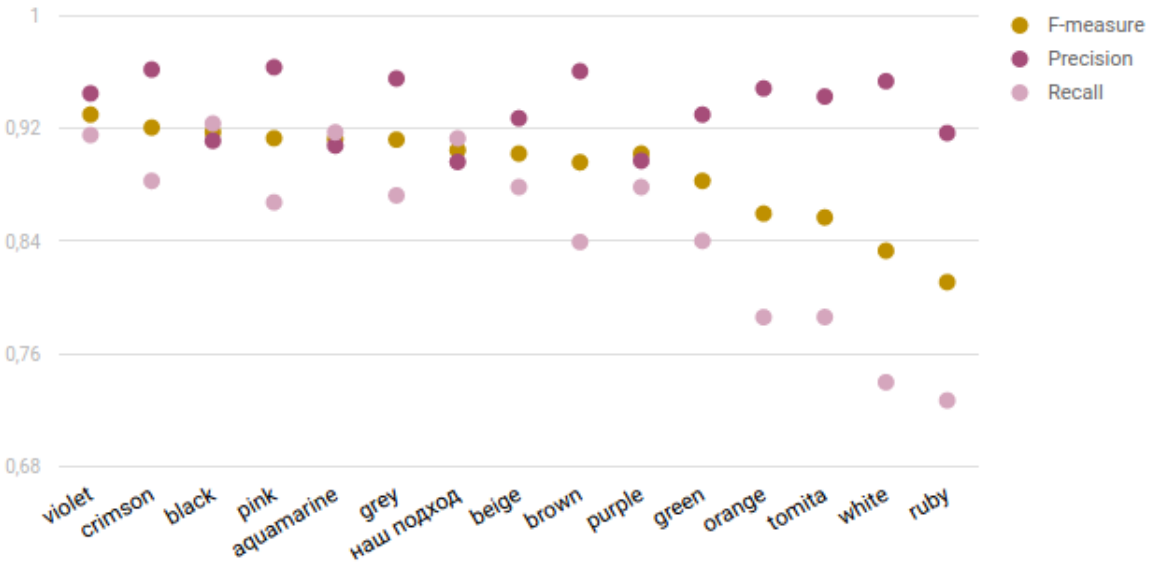

Рис. 3: Распределение результатов извлечения упоминаний

5.2. Результаты этапа кластеризации

Вторая дорожка соревнования была посвящена объединению упоминаний в сущности, к которым они относятся. Только 5 команд приняли участие во второй дорожке.

Результаты сравнения на тестовом множестве представлены в Таблице 6 и на Рис. 4.

	Precision	Recall	F-measure
pink	0.9006	0.8459	0.8724
violet	0.8823	0.8625	0.8723
crimson	0.8712	0.8308	0.8505
aquamarine	0.8162	0.8697	0.8421
наш подход	0.8778	0.8070	0.8409
алгоритм томиты-парсера для кластеризации имён	0.8477	0.8176	0.8324
green	0.7984	0.8419	0.8196

Таблица 6: Результаты кластеризации на тестовом множестве

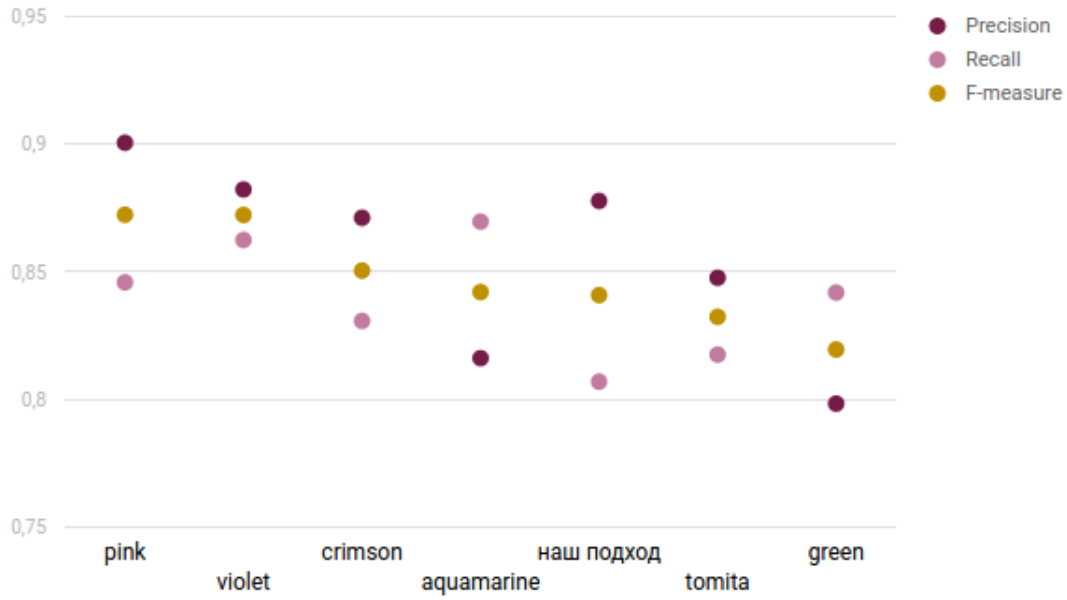

Рис. 4: Распределение результатов кластеризации

5.3. Заключение

В рамках данной работы описан подход для разрешения кореференции для текстов на русском языке.

Разрешение кореференции состояло из двух этапов: извлечение упоминаний и их кластеризация. Извлечение упоминаний происходило при помощи грамматик, написанных для Томита-парсера. Для алгоритма объединения упоминаний была выбрана аггломеративная кластеризация, проходящая на уровне сущностей и использующая взвешенные вектора попарных признаков.

Эксперименты показывают, что мы получили сравнимые результаты, которые оказались выше базового уровня встроенных алгоритмов Томита-парсера. В дальнейшем, мы планируем улучшать результаты за счёт повышения точности на этапе извлечения упоминаний, исправлении ошибок нормализации и усовершенствовании набора признаков. Так же мы планируем перейти к другим типам сущностей (например, локации и организации), другим типам упоминаний (метоимения, имен-

ные группы) и другим типам текстов (например, художественным).

Код проекта выложен в открытом доступе:
https://github.com/lasveritas/coreference-resolution.

Список литературы

[1] ACE (Automatic Content Extraction) English Annotation Guidelines for Entities. — 2008. — URL: https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-entities-guidelines-v6.6.pdf.

[2] Bengtson Eric, Roth Dan. Understanding the Value of Features for Coreference Resolution // Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. — Honolulu, Hawaii : Association for Computational Linguistics, 2008. — October. — P. 294–303. — URL: http://www.aclweb.org/anthology/D08-1031.

[3] Cardie Claire, Wagstaf Kiri. Noun Phrase Coreference as Clustering // 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. — 1999. — P. 82–89. — URL: http://www.aclweb.org/anthology/W99-0611.

[4] Clark Kevin, Manning Christopher D. Entity-Centric Coreference Resolution with Model Stacking // Association for Computational Linguistics (ACL). — 2015.

[5] Coreference Resolution: To What Extent Does It Help NLP Applications? / Ruslan Mitkov, Richard Evans, Constantin Orasan et al. // TSD. — Lecture Notes in Computer Science. — Springer, 2012. — P. 16–27.

[6] Denis Pascal, Baldridge Jason. Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming // Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference. — Rochester, New York : Association for Computational Linguistics, 2007. — April. — P. 236–243. — URL: http://www.aclweb.org/anthology/N/N07/N07-1030.

[7] An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming / Xiaofeng Yang, Jian Su, Jun Lang et al. // Proceedings of ACL-08: HLT. — Columbus, Ohio : Association for Computational Linguistics, 2008. — June. — P. 843–851. — URL: http://www.aclweb.org/anthology/P/P08/P08-1096.

[8] Finkel Jenny Rose, Manning Christopher D. Enforcing Transitivity in Coreference Resolution // Proceedings of ACL-08: HLT, Short Papers. — Columbus, Ohio : Association for Computational Linguistics, 2008. — June. — P. 45–48. — URL: http://www.aclweb.org/anthology/P/P08/P08-2012.

[9] First-order probabilistic models for coreference resolution / Aron Culotta, Michael Wick, Robert Hall, Andrew McCallum // In Proceedings of HLT-NAACL 2007. — 2007.

[10] Grosz Barbara J. The Representation and Use of Focus in a System for Understanding Dialogs // Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1. — IJCAI'77. — 1977. — P. 67–76.

[11] Grosz Barbara J., Joshi Aravind K., Weinstein Scott. Providing a Unified Account of Definite Noun Phrases in Discourse // Proceedings of the 21st Annual Meeting on Association for Computational Linguistics. — ACL '83. — Stroudsburg, PA, USA : Association for Computational Linguistics, 1983. — P. 44–50. — URL: http://dx.doi.org/10.3115/981311.981320.

[12] Grosz Barbara J., Weinstein Scott, Joshi Aravind K. Centering: A Framework for Modeling the Local Coherence of Discourse // Comput. Linguist. — 1995. — . — Vol. 21, no. 2. — P. 203–225.

[13] Haghighi Aria, Klein Dan. Unsupervised Coreference Resolution in a Nonparametric Bayesian Model // Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics. — Prague, Czech Republic : Association for Computational Linguistics, 2007. —

June. — P. 848–855. — URL: http://www.aclweb.org/anthology/P07-1107.

[14] Lavie Alon, Tomita Masaru. GLR* - An Efficient Noise-skipping Parsing Algorithm For Context Free Grammars // In Proceedings of the Third International Workshop on Parsing Technologies. — 1993. — P. 123–134.

[15] Ma Xuezhe, Liu Zhengzhong, Hovy Eduard. Unsupervised Ranking Model for Entity Coreference Resolution // Proceedings of NAACL-2016. — San Diego, California, USA, 2016. — June.

[16] Ma Xuezhe, Liu Zhengzhong, Hovy Eduard. Unsupervised Ranking Model for Entity Coreference Resolution // Proceedings of NAACL-2016. — San Diego, California, USA, 2016. — June.

[17] Mccallum Andrew, Wellner Ben. Toward Conditional Models of Identity Uncertainty with Application to Proper Noun Coreference // In NIPS. — MIT Press, 2003. — P. 905–912.

[18] A Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree / Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing et al. // In Proc. of the ACL. — 2004. — P. 135–142.

[19] Ng Vincent. Machine Learning for Coreference Resolution: From Local Classification to Global Ranking // Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). — Ann Arbor, Michigan : Association for Computational Linguistics, 2005. — June. — P. 157–164. — URL: http://www.aclweb.org/anthology/P05-1020.

[20] Ng Vincent. Unsupervised Models for Coreference Resolution // Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. — Honolulu, Hawaii : Association for Computational Linguistics, 2008. — October. — P. 640–649. — URL: http://www.aclweb.org/anthology/D08-1067.

[21] Ng Vincent. Supervised Noun Phrase Coreference Research: The First Fifteen Years // Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. — Uppsala, Sweden : Association for Computational Linguistics, 2010. — July. — P. 1396–1411.

[22] Ng Vincent, Cardie Claire. Improving Machine Learning Approaches to Coreference Resolution // Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. — ACL '02. — Stroudsburg, PA, USA : Association for Computational Linguistics, 2002. — P. 104–111. — URL: http://dx.doi.org/10.3115/1073083.1073102.

[23] Nicolae Cristina, Nicolae Gabriel. BESTCUT: A Graph Algorithm for Coreference Resolution // Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. — Sydney, Australia : Association for Computational Linguistics, 2006. — July. — P. 275–283. — URL: http://www.aclweb.org/anthology/W06-1633.

[24] Poon Hoifung, Domingos Pedro. Joint Unsupervised Coreference Resolution with Markov Logic. — 2008.

[25] Rahman Altaf, Ng Vincent. Narrowing the Modeling Gap: A Cluster-ranking Approach to Coreference Resolution // J. Artif. Int. Res. — 2011. — . — Vol. 40, no. 1. — P. 469–521. — URL: http://dl.acm.org/citation.cfm?id=2016945.2016958.

[26] Soon Wee Meng, Lim Daniel Chung Yong, Ng Hwee Tou. A Machine Learning Approach to Coreference Resolution of Noun Phrases // Computational Linguistics. — 2001. — Vol. 27, no. 4. — P. 521–544. — URL: http://www.aclweb.org/anthology/J01-4004.

[27] Strube Michael, Rapp Stefan, Müller Christoph. The Influence of Minimum Edit Distance on Reference Resolution // IN PROCEEDINGS OF EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING CONFERENCE. — 2002. — P. 312–319.

[28] Walker Marilyn, Joshi Aravind ans Prince Ellen. Centering Theory in Discourse. — Oxford University Press, 1998.

[29] Wikipedia. Eastern Slavic naming customs — Wikipedia, The Free Encyclopedia. — 2016. — [Online; accessed 24-April-2016]. URL: https://en.wikipedia.org/w/index.php?title=Eastern_Slavic_naming_customs&oldid=715913441.

[30] Wikipedia. Proper noun — Wikipedia, The Free Encyclopedia. — 2016. — [Online; accessed 24-April-2016]. URL: https://en.wikipedia.org/w/index.php?title=Proper_noun&oldid=712472074#Proper_names.

[31] Yang Xiaofeng, Su Jian. Coreference Resolution Using Semantic Relatedness Information from Automatically Discovered Patterns // Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics. — Prague, Czech Republic : Association for Computational Linguistics, 2007. — June. — P. 528–535. — URL: http://www.aclweb.org/anthology/P07-1067.

[32] factRuEval2016. — 2016. — URL: https://github.com/dialogue-evaluation/factRuEval-2016/tree/master/devset.

[33] Томита-парсер. — Технологии Яндекса. — URL: https://tech.yandex.ru/tomita/?ncrnd=2315.

Приложение А Словари словарных слов

Женские обращения:
госпожа, гражданка, графиня, донна, княгиня, княжна, королева, мисс, миссис

Мужские обращения:
господин, гражданин, граф, дон, император, князь, король, лорд, мистер, сэр, царь

Общения с двойным родом:
величество, высочество, генерал, доктор, капитан, майор, полковник, преподаватель, профессор, товарищ

Дескрипторы локаций:
аэропорт, гора, город, губерния, дворец, зона, местность, море, область, озеро, остров, площадь, республика, страна, территория, храм

Дескрипторы организаций:
агентство, альянс, ассамблея, ассоциация, банк, бюро, ведомство, завод, институт, комиссия, комитет, компания, министерство, организация, партия, пресс, пресса, профсоюз, служба, совет, союз, станция, суд, театр,университет, центр

Приложение В Грамматика для извлечения несловарных слов

encoding ”utf-8”
GRAMMAR_ROOT root
define ALL_CASES [nom,acc,dat,ins,loc,gen]

//extra
Conj -> ”и” | Comma;
OrgBegin -> ’ооо’ | ’зао’ | ’оао’;

//name parts
BaseWord -> Word<gram=”S, sg”, kwset= [’names_not’], h-reg1, h-reg2, quoted, l-quoted>;

FirstName -> BaseWord<gram=”persn”>;
FirstName -> BaseWord<gram =”persn, nom, pl, gen, f,m”> interp (RawName .StrongNom=true);
Nob_part -> Word<kwtype=”nob_part”>;
Surname -> (Nob_part) BaseWord<gram=”famn”>;
SpSurname -> AnyWord<wff=”[А-Я].+швили”, dict>;
SpSurname -> AnyWord<wff=”[А-Я].+дзе”, dict>;
SpSurname -> AnyWord<wff=”аль-[А-Я].+”, dict>;

Patronymic -> BaseWord<gram=”patrn”>;

Name -> FirstName | Surname | Patronymic | SpSurname interp (RawName.Name; RawName.Name_not_norm::not_norm; RawName.Pers_check=true);
FOA -> Word<kwtype=”address”, gram = ”sg”>;
FOA_interp -> FOA<gram=”nom”, GU= $ALL_CASES> interp (RawName. StrongNom=true);
Anim -> Word<l-reg, gram = ”S, sg, anim, persn, patrn, famn ”>;

NotAnim -> Word<l-reg, gram = ”S, sg, anim, persn, patrn, famn ”>;

UnknownWord0 -> AnyWord< dict, quoted>;
UnknownWord1 -> UnknownWord0< l-quoted>;
UnknownWord -> UnknownWord1< r-quoted>;
UCName -> UnknownWord<wff=/[А-ЯЁ][а-яё]+/>;
UCName -> UnknownWord<wff=/[А-ЯЁ][а-яё]+-[А-ЯЁ][а-яё]+/>;
DictName0 -> BaseWord<gram=” geo”>;
DictName1 -> DictName0<gram=” famn”>;
DictName -> DictName1<gram=” persn”>;

NotNounName0 -> Word< fw, h-reg1, h-reg2, quoted, wff=’[А-Я][а-я]+’,
gram=” persn”>;
NotNounName1 -> NotNounName0< l-quoted, gram=” famn”>;
NotNounName2 -> NotNounName1< r-quoted, gram=” patrn”>;
NotNounName -> NotNounName2<gram=” geo”> interp
(RawName.Name::not_norm; RawName.Name_not_norm::not_norm);

DictOrNot -> DictName | UCName;

//miscellaneous context
AdjPlace -> Word<kwtype=”adjplace”>;
AdjNESW -> ”южный” | ”западный” | ”северный” | ”восточный”;
LocNameTrue -> DictOrNot AdjPlace<cut>;
LocNameTrue -> AdjPlace<cut> DictOrNot;
LocNameTrue -> AdjPlace<cut> DictName UCName;
LocNameTrue -> AdjPlace<cut> Word<h-reg1> UCName;
LocNameTrue -> AdjNESW<gnc-agr[1]> Word<h-reg1, gnc-agr[1]>;
LocNameTrue -> AdjNESW UCName;
LocNameFalse -> Word<h-reg1, gnc-agr[1]> AdjPlace<gnc-agr[1]> DictOrNot;

LocNameTrue -> ”в”<cut> LocNameTrue;
LocNameTrue -> ”в”<cut> UCName ;

LocTrue_interp -> LocNameTrue interp (RawName.Name;
RawName.Name_not_norm::not_norm;RawName.Misc_check=true);
LocTrue_interp -> LocNameFalse;

OrgTrue -> OrgBegin<cut> UCName;
OrgTrue -> Word<kwtype="org", cut> UCName;
OrgTrue -> UCName Word<kwtype="org", cut>;
OrgTrue -> "представитель"<cut> DictOrNot;
OrgTrue -> Anim<cut> DictName<gram="gen"> FirstName<cut>
Surname<cut>;
OrgTrue -> Anim<cut> UCName FirstName Surname<cut>;
OrgTrue -> Anim<cut> DictName<gram="gen"> Surname<cut>
FirstName<cut>;
OrgTrue -> Anim<cut> UCName Surname<cut> FirstName<cut>;

BareOrg_interp -> DictOrNot interp (RawName.Name;
RawName.Name_not_norm::not_norm;RawName.Misc_check=true);
BareOrg_interp -> NotNounName interp (RawName.Name;
RawName.Name_not_norm::not_norm;RawName.Misc_check=true);
OrgTrue_interp -> OrgTrue interp (RawName.Name;
RawName.Name_not_norm::not_norm;RawName.Misc_check=true);

//person context
DictName_interp -> BaseWord<GU= [geo, persn, famn, patrn]> interp
(RawName.Name;
RawName.Name_not_norm::not_norm);
UnknownName_interp -> UCName interp (RawName.Name;
RawName.Name_not_norm::not_norm);

PersnName_interp -> UnknownName_interp interp
(RawName.Pers_check=true);
NotNounName_interp -> NotNounName interp

(RawName.Pers_check=true);

PersNameTrue -> NotNounName_interp Name;
PersNameTrue -> Name NotNounName_interp;
PersNameTrue -> Name NotNounName_interp NotNounName_interp;
PersNameTrue -> Name NotNounName_interp UnknownName_interp;
PersNameTrue -> FOA NotNounName_interp;

PersNameTrue -> Name PersnName_interp+;
PersNameTrue -> Anim PersnName_interp PersnName_interp;
PersNameTrue -> Anim Name PersnName_interp;
PersNameTrue -> Anim Word<h-reg2> Name PersnName_interp;
PersNameTrue -> Anim Word<h-reg2> PersnName_interp PersnName_interp;

PersNameTrue -> Anim LocTrue_interp Name PersnName_interp;
PersNameTrue -> Anim Word<gram="geo"> Name PersnName_interp;
PersNameTrue -> Anim OrgTrue_interp Name PersnName_interp;
PersNameTrue -> Anim BareOrg_interp Name PersnName_interp;

PersNameTrue -> Anim LocTrue_interp PersnName_interp PersnName_interp;
PersNameTrue -> Anim Word<gram="geo"> PersnName_interp PersnName_inte
PersNameTrue -> Anim OrgTrue_interp PersnName_interp PersnName_interp;
PersNameTrue -> Anim BareOrg_interp PersnName_interp PersnName_interp;

PersNameTrue -> Anim LocTrue_interp Name Name;
PersNameTrue -> Anim Word<gram="geo"> Name Name;
PersNameTrue -> Anim OrgTrue_interp Name Name;
PersNameTrue -> Anim BareOrg_interp Name Name;

PersNameTrue -> PersnName_interp (Comma) Anim;
PersNameTrue -> FOA PersnName_interp;
PersNameTrue -> PersnName_interp (Adv) Verb<gram="sg">;
PersNameTrue -> Verb<gram="sg"> PersnName_interp;

```
PersNameTrue -> Word<gnc-agr[1]> Verb<gram="sg", gnc-agr[1]> UCName;
//false PersNameTrue -> Word<kwtype="hyph"> PersnName_interp Comma;
PersNameTrue -> Comma PersnName_interp Comma;
PersNameTrue -> Comma PersnName_interp EOSent;
PersNameTrue -> PersnName_interp Conj PersnName_interp;

BareUCName -> UnknownWord<wff=/[А-ЯЁ][а-яё]+/,  l-quoted> ;
BareUCName2 -> BareUCName< r-quoted,  fw> interp (RawName.Name;
RawName.Name_not_norm::not_norm);
BareUCName2 -> NotAnim UCName interp (RawName.Name; RawName.Name_
RawName.Misc_check=true);

root -> LocTrue_interp | PersNameTrue | OrgTrue_interp | BareUCName2;
```

Приложение C Грамматика для извлечения полных имён

```
encoding "utf-8"
GRAMMAR_KWSET ["places"]
GRAMMAR_ROOT root

NotPlace -> Word <h-reg1, GU= ["persn, famn, patrn, CONJ, PR, PART,
INTJ"], kwset= ["stop","adjplace","misc", "places", "address"],  quoted>;

~Nob_part -> Word <kwtype = "nob_part">;

ShortName -> Word<wff=/[А-Я]/>;
UCName -> NotPlace<kwtype="names"> outgram = "sg";

FirstName -> NotPlace<gram="persn"> | ShortName;
FamilyName -> NotPlace<gram="famn">;
FamilyName -> Word<wff="аль-[А-Я][а-я]+", kwtype="names">;
FamilyName -> AnyWord<wff="[А-Я].+швили", kwtype="names">;
FamilyName -> AnyWord<wff="[А-Я].+дзе", kwtype="names">;
FamilyName -> Nob_part FamilyName;
PatrnName -> NotPlace<gram="patrn"> | ShortName;

FirstName_interp -> FirstName interp (+StrictName.Firstname);
FamilyName_interp -> FamilyName interp (+StrictName.Lastname);
PatrnName_interp -> PatrnName interp (+StrictName.Patronymic);
UCName_Name_interp -> UCName interp (+StrictName.Firstname);
UCName_Surname_interp -> UCName interp (+StrictName.Lastname);

FirstName_Unl_interp -> FirstName interp (+StrictName.Unlabeled);
FirstName_fam_interp -> FirstName interp (+StrictName.Lastname);
FamilyName_Unl_interp -> FamilyName interp (+StrictName.Unlabeled);
```

```
UCName_Unl_interp -> UCName interp (+StrictName.Unlabeled);
Unl_interp -> FirstName_Unl_interp | FamilyName_Unl_interp |
UCName_Unl_interp;

FullName -> FirstName_interp | FamilyName_interp | PatrnName_interp;
FullName -> FirstName_interp<rt> PatrnName_interp (FamilyName_interp);
FullName -> FamilyName_interp FirstName_interp<rt> (PatrnName_interp);
FullName -> FirstName_interp<rt> FamilyName_interp;
FullName -> FirstName_interp<rt> UCName_Surname_interp;
FullName -> UCName_Name_interp<rt> UCName_Surname_interp;
FullName -> UCName_Name_interp FamilyName_interp;
FullName -> FirstName_interp<rt> FirstName_fam_interp;
FullName -> Unl_interp<GU= [geo, persn, famn]>+;

FullName_interp -> FullName interp (StrictName.NameNotNorm::not_norm);

FOAMale -> Word<kwtype="addressMale"> interp (StrictName.Gender="male");
FOAFemale -> Word<kwset="addressFemale">
interp (StrictName.Gender="female");
FOAFemale -> Word<kwtype="addressFemale_ledi">
interp (StrictName.Gender="female");
FOADual -> Word<kwtype="addressDual"> interp (StrictName.Gender="dual");
FOA_nointerp -> FOAMale | FOAFemale | FOADual;
FOA -> FOA_nointerp interp (StrictName.Address);

FOAName -> FOA FullName_interp;
FOAName -> FOA "и" FOA FullName_interp;

//descriptor
NotName -> Word<gram=" persn,  famn,  patrn", kwset= ['names']>;

BareDescr -> Noun<gram="sg, anim", rt> (Adj<gnc-agr[1]>)
(NotName<gram="gen", gnc-agr[1]>);
```

```
BareDescr -> Noun<gram="sg, anim", rt> NotName<h-reg>;

Descr -> BareDescr<gram="m,  f"> interp (StrictName.Gender="male");
Descr -> BareDescr<gram=" m, f"> interp (StrictName.Gender="female");
Descr -> BareDescr<gram="m, f"> interp (StrictName.Gender="dual");
Descr -> BareDescr interp (StrictName.Gender="dual");

DescrName -> Descr FullName_interp;
DescrName -> Descr<gnc-agr[1]> "-" FullName_interp<gnc-agr[1]>;
DescrName -> Descr<gnc-agr[1]> Hyphen FullName_interp<gnc-agr[1]>;
DescrName -> FullName_interp<gnc-agr[1]> Comma Descr<gnc-agr[1]> ;

DescrName_interp -> DescrName interp (StrictName.Descriptor="true");

//False Examples
Loc -> Word<kwtype="adjplace">;
Org -> Word<kwtype="org">;
MiscTrue -> Loc Word<h-reg1>;
MiscTrue -> Org Word<h-reg1>;
MiscTrue -> "в" Word<h-reg1>;

root -> FullName_interp | FOAName | DescrName_interp | MiscTrue;
```