УДК 800, 801
ББК 81.00

В.А. Яцко

АЛГОРИТМЫ И ПРОГРАММЫ АВТОМАТИЧЕСКОЙ ОБРАБОТКИ ТЕКСТА

Дается обзор наиболее распространённых алгоритмов и программ автоматической обработки текста. Описываются особенности алгоритмов и программ, применяемых на морфологическом, лексическом, синтаксическом и дискурсивном уровнях языковой системы.

Ключевые слова: *автоматическая обработка текста; программы и алгоритмы; морфологический; лексический; синтаксический; дискурсивный уровни языковой системы*

V.A. Yatsko

ALGORITHMS AND PROGRAMS FOR AUTOMATIC TEXT PROCESSING

The paper presents a review of the most widespread algorithms and programs for automatic text processing. Specific features of algorithms and programs classified into morphological, lexical, syntactic, and discursive levels are described.

Key words: *natural language processing; algorithms and programs; morphological; lexical; syntactic; and discursive levels of language*

Особенностью развития лингвистики в настоящее время является тесная взаимосвязь с предметной областью, которая в зарубежной науке получила название *natural language processing* – NLP [Jurafsky, 2009]. В рамках NLP разрабатываются и применяются алгоритмы обработки единиц естественного языка (лингвистические алгоритмы), которые, как мы полагаем, можно классифицировать по таким критериям, как способ коммуника-

ции, форма речи, уровень интеллектуальности, уровень языковой системы.

По способу коммуникации лингвистические алгоритмы можно разделить на два вида: алгоритмы анализа письменного текста и устной речи. Алгоритмы анализа текста разрабатываются с 50-х гг. XX в. и лежат в основе функционирования информационно-поисковых систем, систем автоматического реферирования [Яцко, 2007]. Алгоритмы анализа устной речи стали разрабатываться и широко внедряться в 90-х гг. XX в. и сейчас широко применяются в автоответчиках [Experiences with commercial, 2004], системах распознавания таких индивидуальных характеристик личности, как возраст, пол [Age and gender recognition, 2010] и даже уровень алкогольного опьянения [Use of prosodic speech, 2001], в системах голосового управления техническими объектами [Потапова, 1997], в том числе и наносистемами [Потапова, 2007].

По форме речи можно выделить алгоритмы, предназначенные для обработки монологической и диалогической речи. Долгое время объектом автоматического анализа текста были монологические тексты, в основном тексты научных работ. Развитие интернета обусловило появление жанров диалогической речи: чатов, блогов, форумов. Обработка таких текстов имеет свою специфику и требует применения специальных алгоритмов, учитывающих их паралингвистические особенности [Чухарев, 2008, с. 76-78].

По степени интеллектуальности можно выделить в отдельную группу алгоритмы, с помощью которых пользователю выдаётся информация, содержащаяся в тексте имплицитно, либо новая информация, которой нет в обрабатываемом тексте, например, коэффициенты, отражающие интенсивность какого-либо события. Такие алгоритмы разрабатываются в процессе интеллектуального анализа текста (*text mining*) и существенно отличаются от традиционных алгоритмов информационного поиска и реферирования, в результате применения которых выявляется наиболее значимая информация, содержащаяся в тексте.

Алгоритмы автоматической обработки текста могут применяться на разных уровнях языковой системы, начиная от отдельного символа, который выступает объектом анализа в оптических системах распознавания текста (optical character recognition – OCR) [OCR Systems, 1994], заканчивая дискурсивным уровнем, на котором происходит моделирование структуры связного текста [Marcu, 1999].

В настоящей статье на основе нашего опыта по разработке программного обеспечения для автоматического анализа английских текстов описываются алгоритмы и программы, используемые на разных уровнях языковой системы: морфологическом, лексическом, синтаксическом, дискурсивном.

Алгоритмы морфологического анализа. С помощью алгоритмов морфологического анализа распознаются элементы морфологической структуры слова – корни, основа, аффиксы, окончания. К алгоритмам, широко применяемым на морфологическом уровне, относятся стемминг и лемматизация.

Цель стемминга – отождествить основы семантически схожих словоформ, что необходимо для адекватного взвешивания терминов в процессе информационного поиска. На входе стеммера – текст, на выходе – список основ слов входного текста. Стеммеры, разрабатываемые с конца 50-х гг. XX в., классифицируются на алгоритмические и словарные [Hull, 1996]. Алгоритмические стеммеры функционируют на основе файлов данных, содержащих списки деривационных суффиксов и флексий. В процессе морфологического анализа программа выполняет сопоставление суффиксов и окончаний слов во входном тексте и в соответствующем списке, причём анализ начинается с последнего символа слова. Словарные стеммеры функционируют на основе словарей основ слов. В процессе морфологического анализа такой стеммер выполняет сопоставление основ слов во входном тексте и в соответствующем словаре, а анализ начинается с первого символа слова.

Словарные стеммеры обеспечивают большую точность поиска, в то время как алгоритмические – большую полноту, допуская больше ошибок, которые проявляются в недостаточном или избыточном стеммировании. Избыточное стеммирование (overstemming) имеет место в том случае, если по одной основе отождествляются слова с разной семантикой; при недостаточном стеммировании (understemming) по одной основе не отождест-

ляются слова с одинаковой семантикой. Например, ланкастерский стеммер выделяет *bet* как основу *better*, а *child* – как основу *children*. В первом случае имеет место избыточное стеммирование, поскольку по основе *bet* прилагательное *better* отождествляется с глаголом *bet* и его производными (*bets, betting*), значение которых не имеет ничего общего со значением прилагательного. Во втором случае имеет место недостаточное стеммирование, так как по основе *child* нельзя отождествить формы множественного (*children*) и единственного числа (*child*) одной лексемы.

Несмотря на указанные недостатки, алгоритмические стеммеры намного более распространены, чем словарные. Это объясняется тем, что количество суффиксов и флексий в каждом конкретном языке достаточно ограничено, следовательно, применения на уровне морфологической структуры происходят намного медленнее, чем на лексическом уровне. Стремительное социальное и технологическое развитие обусловливает выход из обращения одних слов и появление других. В первую очередь это относится к существительным, которые создаются для обозначения новых объектов. В британском национальном корпусе, например, нет таких терминов, как *ipod* или *iphone*, поскольку он охватывает тексты, произведенные с 1980-х гг. по 1993 г. Другой проблемой при использовании словарных стеммеров является большой размер словаря, который отрицательно влияет на быстродействие системы. Сферой применения этого вида стеммеров могут быть достаточно узкие предметные области.

В настоящее время наиболее известными стеммерами для английского языка являются алгоритмические стеммеры: стеммер Портера и ланкастерский стеммер, который, по фамилиям разработчиков (C.D. Paice и G. Husk), также называют Paice / Husk stemmer [Paice, 1990]. М. Портер разработал язык программирования Snowball, специально предназначенный для создания стеммеров [Porter, 2001]. Нами был разработан Y-stemmer, который выполняет морфологический анализ на основе предварительного аннотирования тегами частей речи. Это позволяет учитывать только суффиксы и окончания, соотносящиеся с той частью речи, к которой относится данное слово. Кроме того, в Y-стеммере предусмотрено отождествление неправильных форм глаголов, а также существительных и местоимений, образующих множественное число не по правилам [Алгоритмы предварительной обработки текста, 2009].

Эффективность функционирования программ морфологического анализа определяется понятием мощности (strength) стеммера, которая измеряется по соотношению количества словоформ исходного текста и основ слов, оставшихся после стемминга, а также по количеству символов, которые содержатся в удаляемых суффиксах и окончаниях. Например, мощность Y-стеммера, протестированная на известном тексте *The Rubaiyat of Omar Khayyam* по формуле компрессии индекса, (N-S) / N = (1203-1083) / 1203 = 0,09975, где N – количество уникальных словоформ в исходном тексте, а S – количество уникальных основ после стемминга. Мощность стеммера Портера для того же текста равна (1203-1080) / 1203 = 0,10224, т.е. стеммер Портера мощнее Y-стеммера на 0,00249[7]. Данные для ланкастерского стеммера: (1203-1081) / 1203 = 0,10141. Заметим, что нет прямой зависимости между мощностью стеммера и его качеством. Проведенное нами тестирование показало, что качество Y-стеммера на 9,83 % выше, чем качество ланкастерского стеммера [Алгоритмы предварительной обработки текста, 2009].

Лемматизация также предусматривает отождествление основ слов, однако проводится с учетом частей речи, к которым относятся словоформы. Например, стеммер отождествит *read, reads, reader, readers* с одной основой *read*, в то время как лемматайзер отождествит глагольные формы *read, reads* с основой *read*, а именные формы *reader, readers* – с лексемой *reader*. Под лемой понимается лексема, задача лемматизации – отождествить словоформы, соотносящиеся с одной лексемой.

В целях автоматической обработки английских текстов широко применяются лемматизированные и нелемматизированные списки, составленные А. Килгариффом на основе Британского национального корпуса [Kilgarriff,

[7] Данные по стеммеру английской версии стеммера Портера: http://qaa.ath.cx/porter_js_demo.html и Y-стеммера: http://ystemmer.ru/ (прим. ред.)

1997]. Данные списки – это словари, в которых дается информация о частотностях и распределении лемм и словоформ по текстам и частям речи. Эти данные могут использоваться для определения вероятностных характеристик лексических единиц, необходимых, например, для автоматического аннотирования тегами частей речи.

Словари чаще широко используются в корпусной лингвистике в целях поддержки лингвистических исследований. Распределение по частям речи также является существенным параметром при проведении автоматической классификации и категоризации текстов [Santini, 2006].

Алгоритмы лексического анализа. Основная задача лексического анализа – распознать лексические единицы текста. На входе у программ этого типа – текст, на выходе – список лексических единиц текста. Одним из фундаментальных алгоритмов лексического анализа является лексическая декомпозиция, которая предусматривает разбивку текста на токены, соответственно, программы, выполняющие лексическую декомпозицию, называются токенайзерами. Как правило, токены совпадают со словоформами, однако для обозначения лексических единиц текста используется термин «токен», а не «слово», так в ряде случаев под токеном могут пониматься единицы меньше, чем слово (отдельные морфемы, клитики) или больше, чем слово (словосочетания).

Токенайзеры, разработанные для английского языка, выполняют декомпозицию на основе пробелов между словами и обычно распознают в качестве отдельных токенов апостроф и идущие за ним символы (*'s, 'll, 'd, 'm, 're, 've*), от слов отделяются и удаляются знаки пунктуации [Tokenizer, 2010]. Проблема распознавания словосочетаний и аббревиатур решается применением регулярных выражений. Очевидно, что сокращения типа *e.g.* следует распознавать как один токен, то же самое относится к датам, например 11.11.2011. Отдельную проблему представляют инициалы и сокращения перед личными именами, например, *J. Smith, Dr. Smith, J.B. Smith*. Если данные имена являются кореферентными, то имеет смысл рассматривать инициалы как отдельные токены – это позволит распознать *Smith* как имя одного из персонажей и начислить адекватные весовые коэффициенты в зависимости от его частотности. Если же имеются в виду разные люди, следует рассматривать фамилию и инициалы как один токен. Обычно лексическая декомпозиция проводится на основе списков сокращений [Leech, 2000]. Кроме того, в отдельном файле собираются устойчивые словосочетания и идиомы, которые распознаются как один токен. Например, *because of* целесообразно рассматривать как один токен, поскольку это сокращенное словосочетание выражает одно значение.

Лексическая декомпозиция имеет фундаментальное значение для проведения автоматического анализа текста, поскольку лежит в основе целого ряда других алгоритмов. Очевидно, что для проведения стемминга следует вначале разбить текст на токены; на основе списка токенов обычно выполняется синтаксическая декомпозиция, взвешивание, наконец, аннотирование, также выполняемое на лексическом уровне.

Аннотирование проводится теггерами, на входе у которых – список токенов, на выходе – список, в котором каждому токену приписывается условное обозначение (тег), указывающее на его лингвистические характеристики. Наиболее распространенным видом теггеров являются теггеры частей речи (POS taggers), которые распознают часть речи токена и приписывают ему соответствующий тег. Помимо информации о части речи обычно указывается и информация о лексико-грамматических и семантических характеристиках слова, например, *NV* – нарицательное существительное в единственном числе, *NVS* – нарицательное существительное во множественном числе, *AJC* – прилагательное в сравнительной степени и т. д. Списки тегов частей речи различаются по степени дробности. В Британском национальном корпусе используется 58 тегов [BNC part of speech tags, 2008], а в проекте Penn Treebank – 36 [The Penn Treebank Tag Set, 2009]. Более дробная классификация позволяет выдавать пользователю больше информации, однако обусловливает и большее количество ошибок, снижая быстродействие программы.

Теггеры частей речи последовательно выполняют три основных операции: токени-

* http://www.lsi.upc.es/~nlp/freeling/demo.php

цию, морфологическую классификацию и снятие неоднозначности (disambiguation). Морфологическая классификация предусматривает сопоставление каждого токена входного текста со словарем и приписывание ему тегов частей речи. В словаре обычно содержатся словоформы с возможными тегами частей речи. Достаточно большое количество слов соотносится только с одной частью речи (предлоги, артикли, местоимения), однако целый ряд слов может использоваться в качестве различных частей речи. Для английского языка типична омонимия глагольных и именных форм. *Control* может использоваться и как глагол, и как существительное, причем по данным нелемматизированного списка словоформ Британского национального корпуса как существительное в единственном числе (NN1) эта словоформа используется 18 932 раз в 2 318 текстах, как инфинитив (VVI) 3 578 раз в 1 309 текстах, как личная форма (VVB, без учёта использования в третьем лице единственного числа) 957 раз в 559 текстах. К окказиональным относится возможность употребления этой словоформы в качестве прилагательного – 1 случай в 1 тексте. Данная статистическая информация важна для последующей обработки на стадии снятия неоднозначности.

Если какое-либо слово из входного текста отсутствует в словаре, применяются специальные правила для распознавания части речи, к которой оно относится. Если, например, слово оканчивается на *-ious*, то ему приписывается тег прилагательного, поскольку такое окончание типично для английских прилагательных; словам, которые начинаются с большой буквы, приписывается тег собственного имени. В том случае, если невозможно применить правила, токену приписывается тег, используемый по умолчанию, обычно – тег существительного. Существительные – наиболее частотная часть речи знаменательных слов, и именно они обозначают новые объекты, имена которых могут отсутствовать в словаре. Если всем словам в тексте приписать теги существительных, то можно правильно проаннотировать 14.6 % слов [NLTK, 2011].

Токены, которым приписано более одного тега, а также статистическая информация о них передаются для дальнейшей обработки в модуль снятия неоднозначности. Снятие неоднозначности предусматривает выбор одного из двух или более тегов, приписанных данному токену. В зависимости от алгоритмов, применяемых для снятия неоднозначности, теггеры частей речи классифицируются на стохастические и основанные на правилах (*rule-based*). В стохастических теггерах проводится анализ вероятностных параметров каждого из тегов (обычно на основе скрытых марковских моделей), в результате которого выбирается один тег с наибольшим вероятностным значением. Распространен алгоритм двунаправленной инференции, выполнение которого предусматривает анализ тегов токенов справа и слева от текущего токена [Tsuruoka, 2005].

В теггерах, основанных на правилах, анализ вероятностных характеристик не проводится, хотя учитываются частотности использования тегов с тем или иным токеном. Такой теггер обучается на достаточно большом аннотированном корпусе, запоминая наиболее частотные теги морфологически омонимичных словоформ, далее, для повышения качества аннотирования применяются специальные правила автоматического исправления ошибок (*patching rules*). В качестве примера можно привести теггер, разработанный Эриком Бриллом [Brill, 1992], который обучался на Брауновском корпусе, содержащем более миллиона словоформ. При настройке теггера использовалось три группы правил: правила, учитывающие лексические параметры текущего токена, правила, учитывающие контекст токена, правила, учитывающие расстояние от текущего токена до другого токена с определенным лексическим параметром. Обучение теггера проводилось на 90 % текстов корпуса, 5 % использовались для тестирования и распознавания ошибок, аннотация теггера сравнивалась с аннотацией корпуса, на оставшихся 5 % оценивалась эффективность правил. Без применения правил теггер допускал 7.9 % ошибок, а после применения и доработки правил количество ошибок снизилось до 3.5 %. В Британском национальном корпусе количество ошибок составляет 1.5 %, причем при аннотировании используется гибридная технология с использованием стохастического теггера и модифицирующих правил [Leech, 2000]. Интересно, что в данном корпусе при-

меняется технология бинарных тегов: если для токена не удалось найти один тег, то ему приписывается два тега, первый из которых является наиболее вероятным, например, *A/O-A/O* указывает на то, что более вероятным является тег прилагательного. Окончательный выбор тега оставляется на усмотрение пользователя.

В целом стохастические и гибридные технологии существенно снижают количество ошибок, однако отрицательно влияют на быстродействие системы. Их можно успешно использовать для аннотирования статических корпусов. Для динамического аннотирования предпочтительнее применять теггеры, основанные на правилах, поскольку они обеспечивают большее быстродействие. Перспективным направлением развития динамического аннотирования является создание фактографических поисковых систем. В настоящее время в таких системах используется аннотирование такими семантическими тегами, как *Person, Location, Organization*.

InFact, одна из таких систем, разработанная в компании *Insightful Corporation*, позволяет получать информацию по запросам типа *[Organization Name] buy [Organization Name]^money*. В ответ на такой запрос пользователю будет выданы клаузы текста, в которых содержится информация о покупке одной компанией другой компании за определённую сумму денег [A case study in natural language based Web search, 2007]. Для аннотирования семантическими тегами используется программное обеспечение Talent, разработанное фирмой IBM [Cooper, 1998].

Еще одним направлением является аннотирование тегами когнитивных ролей (knowledge roles), которое применяется в интеллектуальном анализе текста (text mining). В [Mustafaraj, 2007] проводилось аннотирование текстов диагностических отчетов о состоянии электроизоляции высоковольтных ротационных устройств такими ролями, как *Observed Object, Symptom, Cause*. В результате была создана система, с помощью которой инженер мог получать информацию о признаках неполадки конкретного объекта, причинах и способах её устранения. В качестве лингвистической базы данных использовалась лексикографическая информация, разработанная в рамках проекта Framenet [Baker, 1998].

Следует отметить, что аннотирование семантическими и когнитивными ролями предусматривает распознавание как отдельных слов, так и словосочетаний. Такое аннотирование требует предварительной разработки и применения специальных грамматик фразовой структуры на синтаксическом уровне языковой системы.

Алгоритмы синтаксического и дискурсивного анализа. Одним из фундаментальных алгоритмов, применяемых на синтаксическом уровне, является синтаксическая декомпозиция (syntactic splitting). На входе у сплиттера текст, на выходе – список предложений текста. Алгоритмы синтаксической декомпозиции разрабатываются с 1960-х гг. и предусматривают распознавание предложений на основе символов форматирования текста: пробелов, знаков пунктуации, знаков перевода каретки. Разбивка текста на предложения осложняется отсутствием стандартного форматирования текста: точки, восклицательные, вопросительные знаки, которые обычно применяются в качестве разделителей, могут использоваться не только в конце, но и в середине предложения. Целый ряд единиц текста, которые форматируются как предложения, на самом деле предложениями не являются. К ним относятся такие элементы, как оглавление, заглавия отдельных разделов, названия рисунков, таблиц, текст, использующийся внутри самих таблиц и рисунков, колонтитулы. Между тем именно предложения являются основной единицей анализа во многих системах, а в системах автоматического реферирования и выходной текст состоит из предложений. Ошибки в распознавании предложений существенно снижают эффективность таких систем в целом.

Нами была предложена дедукционно-инверсионная архитектура декомпозиции текста, в соответствии с которой вначале текст разбивается на абзацы, затем – на слова, затем из слов генерируются предложения. Таким образом, декомпозиция начинается с большей единицы (абзаца), затем осуществляется переход к меньшей единице (слову), затем снова к большей (предложению). Дедукционно-инверсионная архитектура декомпозиции позволяет игнорировать такие компоненты

текста, как заголовки, подзаголовки, оглавления, поскольку они не входят в состав абзацев [Разработка методов и алгоритмов повышения эффективности распознавания жанра и адаптивного реферирования текста, 2009].

Синтаксическая декомпозиция является основой для выполнения целого ряда алгоритмов распознавания фразовой структуры предложения. Широко распространены алгоритмы выделения *n-gram* – словосочетаний, состоящих из двух (биграмы), трёх (триграмы) и более (тетраграмы, пентаграмы, гексаграмы, гептаграмы, октограмы) токенов [Bickel, 2005, Zhang, 2003]. Разбивка на словосочетания в данном случае проводится с учетом позиции токена в предложении. Например, предложение *John has a dog* включает 4 юнитграма, 3 биграма (*John has, has a, a dog*), 2 триграма (*John has a, has a dog*), 1 тетраграм – всё предложение. Количество биграм для каждого предложения (ng_{2}) будет составлять n-1, триграм – n-2, где n-количество токенов в предложении, т. е. $ng_{2} = w_{1,2}, w_{2,3}, \ldots w_{n-1,n}$, где w_{i} – порядковый уровень *n-gram*, начиная с биграм. Распознавание *n-gram* проводится на основе соответствующих правил.

Анализ распределения *n-gram* позволяет выявить статистически значимые словосочетания и часто применяется в стохастических алгоритмах аннотирования тегами частей речи. При этом начало и конец предложения обозначаются некоторыми условными тегами (*false tags*), что позволяет рассматривать в качестве триграм даже предложения, состоящие из одного токена и устанавливать вероятностные параметры, необходимые для выбора того или иного тега. В корпусе современного американского варианта английского языка (COCA)* *I like to* встречается 4 810 раз, в то время как *I like it* – 29 раз, что указывает на гораздо большую вероятность первого словосочетания. Распределения *n-gram* используются с целью автоматической классификации и категоризации, поскольку выступают в качестве важного параметра, позволяющего определить принадлежность текста к определенной категории, типу, группе, жанру. При анализе на синтаксическом уровне в качестве основной единицы выступают биграммы и триграммы, поскольку рекуррентность словосочетаний с большим количеством токенов маловероятна. Анализ *n-gram* большего порядка применяется в системах автоматической коррекции орфографии, а также в системах автоматического распознавания текстов (*Optical Character Recognition*), где основной единицей выступают символы и токены.

Для анализа морфологически значимых словосочетаний применяются чанкеры (*chunkers*), которые на выходе выдают списки фраз определенного типа (именные, глагольные, адъективные, адвербиальные). Наиболее распространены именные (noun phrase) чанкеры, распознающие словосочетания с управляющим существительным. Именно этим типом словосочетаний обозначаются объекты, описываемые в тексте, а их ранжирование по весовым коэффициентам позволяет получить список ключевых слов, отражающих основное содержание текста. Как мы показывали ранее [Яцко, 2002], реферирование текста на основе словаря существительных позволяет получить практически такие же результаты, как и реферирование, проводимое с учетом слов, относящихся к другим частям речи. Распознавание словосочетаний этого типа выполняется на основе предварительного аннотирования тегами частей и объединения отдельных частей речи во фразы на основе правил грамматики.

Правила фразовой структуры были разработаны для английского языка в рамках концепции генеративной грамматики, предложенной Н. Хомским. Грамматические правила записываются в виде NP → NN, NP → DetNN, NP → DetANN, где указывается состав словосочетания, в данном случае именного (*noun phrase – NP*), а также порядок слов [Bilton, 2000, p. 168-188]. В первом случае показано, что именное словосочетание может состоять только из одного существительного (*NN*), во втором случае оно состоит из детерминанта (*Det*) и существительного, причем детерминант занимает позицию перед существительным, а обратный порядок слов неправилен; в третьем случае словосочетание состоит из детерминанта, прилагательного (*A*), существительного, при этом другие варианты словопорядка неправильны.

К настоящему времени на основе концепции Н. Хомского создан целый ряд грамматик,

* http://www.americancorpus.org/

которые делятся на два основных вида – деривационные и недеривационные [Vincent, 2009]. В деривационных грамматиках проводится разграничение между поверхностной и глубинной структурой словосочетания и предложения и формулируются дополнительные правила вывода (деривации) поверхностных структур из глубинных. Синтаксическая структура представляется в виде иерархического дерева зависимости. Недеривационные грамматики описывают поверхностные как правило, линейные синтаксические структуры. Выбор того или иного типа грамматики обуславливается задачами конкретного исследовательского проекта.

Деривационные грамматики лежат в основе функционирования синтаксических парсеров (syntactic parsers), которые выдают на выходе граф синтаксической структуры предложения. Так же, как и теггеры частей речи, синтаксические парсеры обучаются на предложениях с размеченной вручную синтаксической структурой, в них применяется правила для определения наиболее вероятного варианта на основе скрытых моделей Маркова. В качестве примера можно привести *Lexparser*, разработанный в Стэнфордском университете США.

Иерархические синтаксические структуры применяются в системах машинного перевода для установления эквивалентности синтаксических структур в двух языках.

На синтаксическом уровне может проводиться декомпозиция не только на словосочетания и предложения, но и на клаузы.

Элементарные предикативные структуры, выражающие суждение. Понятие клаузы в определенной степени соответствует понятию пропозиции в лингвистике, однако клаузы выделяются по формальным признакам, к которым может относиться, например, наличие именной группы и следующей за ней глагольной группы. Разбивка на клаузы применяется в системах интеллектуального анализа для более адекватной передачи содержания текста, например, описанный выше проект немецких исследователей [Mustafaraj, 2007].

Наиболее распространенными алгоритмами, применяемыми на дискурсивном уровне (уровне связного текста), являются алгоритмы разрешения анафоры, которые предусматривают замену анафорических местоимений предшествующими кореферентными именами объектов.

К настоящему времени сложилось два основных подхода к разработке алгоритмов разрешения анафоры: глобально-дискурсивный и статистический. Глобально-дискурсивный подход предполагает распознавание антецедентов на основе моделирования тематической структуры текста, в то время как статистический подход, в рамках которого выполняется большинство исследований, основывается на приписывании возможным антецедентам весовых коэффициентов. В обобщённом виде алгоритм разрешения анафоры в рамках статистического подхода включает три этапа: 1. Просмотр контекста и распознавание по определенным критериям возможных антецедентов для текущего местоимения. В качестве контекста могут выступать словосочетания в данном предложении *i* клаузе, предшествующие предложения *i* клаузы, а также последующие предложения *i* клаузы, если текст начинается с местоимения. 2. Приписывание весовых коэффициентов каждому из возможных антецедентов на основе определённых параметров и факторов. В этом смысле статистические алгоритмы разрешения анафоры можно отнести к методам факторного анализа. 3. Выбор в качестве кореферентного антецедента с наибольшим весом.

В [Lappin, 1994] описывается алгоритм, разработанный для анализа технических текстов. Алгоритм основывается на четырех группах правил: правила фильтрации именных словосочетаний, которые не могут быть антецедентом местоимения, правила распознавания весовых параметров возможных именных словосочетаний-антецедентов, правила ранжирования возможных антецедентов по весовым параметрам, правила выбора наиболее вероятного из возможных антецедентов.

Фильтрация проводится по ряду лексико-грамматических и синтаксических параметров. В предложении *The woman said that he is busy* именная группа *The woman* не может быть антецедентом местоимения *he* поскольку они не согласуются в роде и используются с глаголами в разной временной форме. В *John seems to himself to see him* собственное имя *John* не может быть антецедентом местоимения *him* поскольку оно входит в состав обя-

тельного аргумента глагола, занимающего позицию непосредственно после собственного имени. Для отдельных видов местоимений разрабатываются собственные правила фильтрации.

После проведения фильтрации идентифицируются именные группы, которые могут быть возможными антецедентами местоимения в текущем предложении. Если возможных антецедентов больше одного, то для каждого из них определяются параметры, по которым проводится их взвешивание. Антецедент с большим весом выбирается в качестве референтного термина. К параметрам, от которых зависит весовой коэффициент именной группы, относятся использование в субъектной позиции, позиции прямого дополнения, косвенного дополнения, предложного дополнения, использование в не-адвербиальной позиции, использование в качестве управляющего существительного. Наибольший коэффициент получают уникальные (не повторяющиеся) именные группы. В случае повторного использования коэффициент уникальности и все остальные коэффициенты снижаются в два раза. Применение весовых коэффициентов и алгоритма в целом можно продемонстрировать на примере следующего текста:

(1) You have not waited for the file to close. (2) You may have asked to print on the virtual printer, but it cannot print until the output file is closed.

После фильтрации некореферентных имён остается два возможных антецедента местоимения *it*: *file* в (1) и *printer* в (2). Поскольку *printer* встречается в тексте 1 раз, то ему присваивается коэффициент уникальности 100, а термину *file*, встречающемуся 2 раза – коэффициент 50, причём коэффициенты для всех параметров *file* также снижаются в два раза. Оба существительных используются в неадвербиальной позиции, поэтому *printer* начисляется дополнительный коэффициент 50, а *file* – коэффициент 25. Оба термина являются управляющими существительными, по этому параметру *printer* начисляется коэффициент 80, а *file* – коэффициент 40. Кроме того, *printer* начисляется коэффициент 40 за использование в позиции предложного дополнения, а *file* – коэффициент 40 – за использование в субъектной позиции и коэффициент 35 за использование в параллельных синтаксических ролях в двух предложениях. Общий коэффициент для *printer*=270, для *file*=190, соответственно в качестве референта *it* выбирается *printer*.

Эффективность данного алгоритма была протестирована на корпусе из 48 текстов инструкций для пользователей компьютеров. Из корпуса произвольным образом были отобраны предложения, содержащие хотя бы одно местоимение, а затем для каждого предложения было выписано предшествующее предложение. В целом во всех предложениях оказалось 360 местоимений, и для 310 из них были правильно выбраны антецеденты, что дало качество в 86 % [Lappin, 1994, p. 554].

Очевидно, что для выполнения такого алгоритма необходима достаточно сложная грамматика, с помощью которой распознаются виды фраз, синтаксические роли, лексико-грамматические параметры. При этом разрешение анафоры выполняется с учетом не более чем 4 предшествующих предложений, что существенно ограничивает сферу применения алгоритма.

Алгоритмы разрешения анафоры имеют существенное значение для всех направлений автоматического анализа текста. В системах автоматического реферирования и информационного поиска замена местоимений референтными именами позволяет произвести более адекватное взвешивание терминов и существенно повысить качество результата. Алгоритмы разрешения анафоры важны и для разработки вопросно-ответных систем, систем интеллектуального анализа текста, а также и для систем искусственного интеллекта, поскольку разрешение анафоры предполагает формулирование логического вывода (inference) [Barland, 2009].

Данная статья написана на основе опыта разработки программ автоматического анализа английских текстов в лаборатории компьютерной лингвистики ХГУ им. Н.Ф. Катанова и, разумеется, не претендует на их исчерпывающее описание. Следует также иметь в виду, что предложенная классификация алгоритмов в зависимости от уровней языковой системы является достаточно абстрактной. В реальности описанные алгоритмы могут применяться в разной последовательности. Например, синтаксическая декомпозиция может предшествовать лексической: сначала распознаваться

предложения, а затем выделяться токены, из которых они состоят. В некоторых случаях существует достаточно жёсткая последовательность выполнения алгоритмов, в частности, аннотирование обязательно предусматривает предварительную лексическую декомпозицию. Ряд однотипных алгоритмов может применяться на разных уровнях языковой системы. Наиболее типичный пример – взвешивание терминов: весовые коэффициенты, определяемые по результатам взвешивания, могут приписываться отдельным словам, словосочетаниям, предложениям, группам предложений, а также тексту в целом.

Возможна классификация алгоритмов автоматического анализа текста и по другим критериям. Существенной характеристикой, как мы считаем, является выделение динамических и статических алгоритмов. Динамические алгоритмы выполняются «на лету», в ответ на запрос пользователя, в то время как статические алгоритмы выполняются в процессе предварительного анализа текста, долгого, как к нему обращается пользователь. Соответственно, существенно различаются требования к быстродействию, что в свою очередь влияет на выбор архитектуры и языка используемого программного обеспечения. Наибольшее быстродействие достигается при применении алгоритмов поверхностного уровня, к которым относятся позиционно-статистические алгоритмы. Более сложны алгоритмы семантического уровня, предусматривающие анализ семантики языковых единиц (например, структурно-семантических отношений), либо анализ структуры связного текста, в том числе моделирование его тематической структуры. Таким образом, можно выделить три группы алгоритмов: алгоритмы поверхностного, семантико-синтаксического и дискурсивного уровней. Для поддержки этих алгоритмов используются и разные лексикографические ресурсы. Алгоритмы поверхностного уровня выполняются на основе словарей, содержащих статистико-вероятностные данные о распределении языковых единиц. Для выполнения алгоритмов семантического уровня требуются словари-тезаурусы, семантические словари, онтологии. Перспективным направлением, как нам представляется, выступает разработка словарей, в которых указываются такие семантические признаки слов, выделяемые в компонентном анализе [Brinton, 2000, p. 138-150], как «одушевлённость – неодушевлённость», «абстрактность – конкретность», благодаря которым можно существенно повысить эффективность выполнения, например, разрешения анафоры. Вообще, актуальным является более широкое применение и алгоритмизация таких лингвистических методов, как компонентный, предикационный, падежно-ролевой анализ.

При разработке лингвистического программного обеспечения используются самые разнообразные языки программирования. Наиболее популярны языки группы C (C++, C#). В США широко применяется язык Python, в частности в инструментальном ПО NLTK (Natural Language Toolkit), разработанном в Пенсильванском университете [Mertz, 2004].

Существенная проблема, с которой сталкиваются разработчики современного лингвистического ПО – плохое качество текстов, размещаемых в Интернете. К таким текстам, как чаты, достаточно трудно, а зачастую и невозможно, применить традиционные алгоритмы анализа в силу многочисленных отклонений от норм орфографии, пунктуации и грамматики. Вместе с тем именно такие жанры текстов, как чаты, блоги, форумы служат ценным источником информации и являются объектом анализа в целом ряде областей, в первую очередь в программах интеллектуального анализа текста. Разработка алгоритмов анализа диалогических текстов также является перспективным направлением в рамках автоматической обработки естественного языка.

Библиографический список

1 *Технологии* предварительной обработки текста : декомпозиция, аннотирование, морфологический анализ [Текст] / В.А. Яцко, М.С. Стариков, С.В. Ларченко [и др.] // Научно-техническая информация. Сер.2. – 2009. – № 11. – С. 8-18.

2 *Потапова, Р.К.* Нанотехнологии и лингвистика : прогнозы и перспективы взаимодействия [Текст] / Р.К. Потапова // Нанотехнологии в лингвистике и лингводидактике: миф или реальность? Опыт создания общего образовательного пространства стран СНГ : тезисы междунар. науч.-практ. конф. (Москва, 2007 г.). – М. : МГЛУ, 2007. – С. 9-11.

5. *Потапова, Р.К.* Речь: коммуникация, информация, кибернетика [Текст] / Р.К. Потапова. – М. : Радио и связь, 1997. – 528 с.
4. *Пухарев, Е.М.* Лингвостатистические корреляты спонтанности в компьютерно-опосредованном дискурсе (на материале русскоязычного чата) [Текст] : дис. … канд. филол. наук : 10.02.21 / Е.М. Пухарев. – СПб., 2008. – 211 с.
5. *Ягуно, В.А.* Некоторые проблемы разработки современных систем автоматического реферирования текста [Текст] / В.А. Ягуно, Т.В. Вишняков // Научно-техническая информация. Сер.2. – 2007. – №9. – С. 7-12.
6. *Ягуно, В.А.* Симметричное реферирование: теоретические основы и методика [Текст] / В.А. Ягуно // Научно-техническая информация. Сер.2. – 2002. – №5. – С. 19-28.
7. *A case* study in natural language based Web search [Text] / G. Marchisio, N. Dhillon, J. Liang [et al.] // Natural language processing and text mining / A. Eds Kao, S. Poteet. – London : Springer-Verlag, 2007. – P. 69-90.
8. *Age* and gender recognition based on multiple systems – early vs. late fusion [Text] / T. Bocklet, G. Stemmer, V. Zeissler [et al.] // Proceedings of the 11th annual conference of the international speech communication association. – Makuhari, Chiba, Japan : ISCA, 2010. – P. 2830-2833.
9. *Baker, C.F.* The Berkeley framenet project [Electronic resource] / C.F. Baker, C.J. Fillmore, J.B. Lowe. – 1998. – URL : http://framenet.icsi.berkeley.edu/papers/acl98.pdf (дата обращения : 05.02.2012).
10. *Barland, I.* Propositional Logic: inference rules [Electronic resource] / I. Barland, J. Greiner. 2009. – URL : http://cnx.org/content/m10718/latest/ (дата обращения : 05.02.2012).
11. *Bickel, S.* Predicting Sentences using N-Gram Language Models [Electronic resource] / S. Bickel, P. Haider, T. Scheffer. – 2005. – URL : http://www.mpi-inf.mpg.de/~bickel/publications/bickel_emnlp_2005.pdf (дата обращения : 05.02.2012).
12. *BNC* part of speech tags [Electronic resource] – 2008. – URL : http://pie.usna.edu/POScodes.html (дата обращения : 05.02.2012).
13. *Brill, E.* A simple rule-based part of speech tagger [Text] / E. Brill // ANLC '92 Proceedings of the third conference on Applied natural language processing. – 1992. – P. 112-116.
14. *Brinton, L.J.* The structure of modern English [Text] / L.J. Brinton. – Amsterdam; Philadelphia : John Benjamins, 2000. – 335 p.
15. *Cooper, J.W.* A Visual Interface for prompted query refinement [Electronic resource] / J.W. Cooper, B.R.J. Obivan // IBM Thomas J. Watson research center. – 1998. – URL : http://www.research.ibm.com/talent/documents/cooper_hicss31.pdf (дата обращения : 05.02.2012).
16. *Experiences* with commercial telephone-based dialogue systems [Text] / E. Noth, A. Horndasch, F. Gallwitz [et al.] // Information technology. – 2004. – V. 46. – №6. – P. 315-321.
17. *Hull, D.A.* Stemming algorithms: a case study for detailed evaluation [Text] / D.A. Hull // Journal of the American Society for Information Science. – 1996. – Vol. 47, №1. – P. 70-84.
18. *Jurafsky, D.* Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition [Text] / D. Jurafsky, J.H. Martin. – 2-d ed. – N.J. : Pearson / Prentice Hall; London : Pearson Education, 2009. – 1024 p.
19. *Kilgarriff, A.* Putting frequencies in the dictionary [Text] / A. Kilgarriff // International Journal of Lexicography. – 1997. – V. 10, №2. – P. 135-155.
20. *Lappin, S.* An algorithm for pronominal anaphora resolution [Text] / S. Lappin, H.J. Leass // Computational linguistics. – 1994. – V.20, №4. – P. 535-561.
21. *Leech, G.* Manual to accompany the British National Corpus (version 2) with improved word-class tagging [Electronic resource] / G. Leech, N. Smith. – 2000. – URL : http://ucrel.lancs.ac.uk/bnc2/bnc2postag_manual.htm (дата обращения : 05.02.2012).
22. *Marcu, D.* Discourse trees are good indicators of importance in text [Text] / D. Marcu // Advances in automatic text summarization. – Cambridge; London : The MIT Press, 1999. – P. 123-136.
23. *Mertz, D.* Charming Python : get started with the Natural Language Toolkit [Electronic resource] / D. Mertz. – 2004. – URL : http://www.ibm.com/developerworks/linux/library/l-cpnltk.html (дата обращения : 05.02.2012).
24. *Mustafaraj, E.* Mining diagnostic text reports by learning to annotate knowledge roles / E. Mustafaraj, V. Hoof, D. Freudenberg // Natural language processing and text mining [Text] / A.Eds Kao, S. Poteet. – London, 2007. – P. 45-68.
25. *NLTK* : natural language toolkit : development : Google project hosting. Taggers [Electronic resource]. – 2011. – URL : http://nltk.googlecode.com/svn/trunk/doc/howto/tag.html (дата обращения : 05.02.2012).
26. *O'R* Systems / NCTP Staff [Electronic resource]. – 1994. – URL : http://www2.edc.org/NCIP/LIBRARY/V1/oor.htm (дата обращения : 05.02.2012).
27. *Paice, C.D.* Another stemmer [Text] / C.D. Paice // SIGIR forum. – 1990. – Vol. 24, №3. – P. 56-61.
28. *Porter, M.F.* Snowball : A language for stemming algorithms [Electronic resource] / M.F. Porter. – 2001. – URL : http://snowball.tartarus.org/texts/introduction.html (дата обращения : 05.02.2012).
29. *Santini, M.* Common criteria for genre classification: annotation and granularity [Text] / M. Santini // 3-rd international workshop on text-based information retrieval (TIR-06). – Riva del Garda, Italy : University of Trento, 2006. – P. 35-40.
30. *The Penn* Treebank Tag Set [Electronic resource] // IMS Stuttgart. – 2009. – URL : http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CQPHTML-Demo/PennTreebankTS.html (дата обращения : 05.02.2012).

31. *Tokenizer : Opennlp* [Electronic resource] – SourceForgeNet, 2010. – URL : http://sourceforge.net/apps/mediawiki/opennlp/index.php?title=Tokenizer (дата обращения : 05.02.2012).

32. *Tsuruoka, Y.* Bidirectional Inference with the easiest-first strategy for tagging sequence data [Text] / Y. Tsuruoka, J. Tsujii // Proceedings of HLT/EMNLP. – 2005. – P. 467-474.

33. *Use* of prosodic speech characteristics for automated detection of alcohol intoxication [Text] / M. Levit, R. Huber, A. Batliner [et al.] // Proceedings of the workshop on prosody and speech recognition. – Red Bank, NJ : ISCA, 2001. – P. 103-110.

34. *Vincent, N.* LFG and dynamic syntax : two non-derivational theories [Text] / N. Vincent // Proceedings of the LFG09 conference. – UK : Cambridge, 2009. – P. 587-603.

35. *Zhang, S.* An effective combination of different order n-grams [Electronic resource] / S. Zhang, N. Dong // Proceedings of the 17-th pacific Asia conference on language, information and computation. – Singapore, 2003. – P. 251-256. – URL : http://aclweb.org/anthology-new/Y/Y03/Y03-1028.pdf (дата обращения : 05.02.2012).