А. Е. Поляков

ТЕХНОЛОГИЯ ПОДГОТОВКИ ИНФОРМАЦИИ В НАЦИОНАЛЬНОМ КОРПУСЕ РУССКОГО ЯЗЫКА

1. Введение.

Национальный корпус русского языка представляет собой коллекцию электронных текстов, отражающих разнообразные стили, жанры и варианты русского языка и снабженных обширной лингвистической и метатекстовой информацией. Наличие такой информации является главной особенностью корпуса, отличающей его от обычных собраний текстов, широко доступных в электронной форме, в том числе в Интернете. Детальность и достоверность этой информации, а также широкий охват различных языковых фактов и явлений, представленных в текстах, составляет основную ценность корпуса как уникального лингвистического ресурса.

Корпус содержит информацию двух основных типов:

A. Метатекстовая информация.

Включает признаки, характеризующие текст в целом, такие как: имя автора, его пол, дата рождения, заглавие текста, дата создания текста, объем в словах, тематика, тип текста, жанр, сфера функционирования и др.[1]

B. Лексическая информация.

Включает признаки, характеризующие отдельное словоупотребление, т. е. словоформу в конкретном месте в корпусе текстов. Сюда относятся:

B.1. Морфологические признаки:

- лексема (словарная форма),
- грамматические признаки лексемы (часть речи, одушевленность, переходность),
- грамматические признаки словоформы (число, падеж, наклонение, время, лицо).

[1] Подробнее см. статью С. О. Савчук в настоящем сборнике.

В.2. Семантические признаки:

семантический разряд, таксономический класс, мереология, оценка, каузация, словообразовательные связи и др.[1]

Текст в корпусе рассматривается как последовательность абзацев, которые состоят из предложений, а предложения из слов, причем основной единицей анализа является слово, а основной единицей контекста — предложение. Поиск в корпусе позволяет находить слова и сочетания слов с определенными признаками, но только в рамках предложения. Результатом поиска является список предложений, в которых найденные слова выделены шрифтом. При необходимости поисковый контекст может быть расширен до пределов абзаца, но не более того.

Таким образом, в корпусе выделяются следующие основные структурные единицы: слово, предложение, абзац, текст. Здесь не используются единицы больше абзаца, отражающие структурное деление текста (части, главы, разделы), и единицы, отражающие синтаксическую структуру предложения (клаузы, группы). Информация приписывается только минимальной и максимальной единицам структуры: слову и целому тексту. Некоторые элементы синтаксической разметки присутствуют только для обозначения устойчивых словосочетаний (оборотов).

Корпус подразделяется на две неравные части, которые заметно отличаются объемом и детальностью разметки. Основную часть корпуса составляют тексты, где каждому слову автоматически приписаны все возможные морфологические разборы. В небольшой части корпуса сделана более детальная разметка, а именно: вручную снята морфологическая омонимия, словам приписаны семантические признаки, расставлены акценты в словах. Эта часть корпуса дает наиболее точную информацию и может рассматриваться как некоторый эталон для лингвистических корпусов. Метатекстовая разметка одинакова в обеих частях корпуса.

2. Принципы представления информации в корпусе.

Разработка оптимального формата разметки для корпуса оказалась достаточно сложной задачей как в теоретическом, так и в

[1] Подробнее см. статью Г. И. Кустовой, О. Н. Ляшевской, Е. В. Падучевой, Е. В. Рахилиной «Семантическая разметка лексики в Национальном корпусе русского языка: принципы, проблемы, перспективы» в настоящем сборнике.

практическом плане. Формат кодирования информации для корпуса должен удовлетворять множеству противоречивых требований, из которых укажем наиболее существенные:

1) полнота — возможность представить всю релевантную информацию (лингвистическую и метатекстовую);

2) расширяемость — простота добавления новых видов информации;

3) компактность — отсутствие избыточной разметки, диктуемой только требованиями формата, а не содержанием;

4) понятность — удобство восприятия для человека, которое необходимо для ручного контроля и редактирования разметки;

5) согласованность с используемым программным обеспечением (морфологический парсер, поисковый движок, фильтры);

6) легкость конвертирования в другие форматы, возможность автоматизации обработки;

7) прозрачность — полное сохранение исходного текста при наложении разметки. После снятия этой разметки должен восстанавливаться исходный текст без каких-либо искажений.

2.1. Существующие стандарты и технологии разметки корпусов.

Большинство современных языков разметки основано на SGML/XML, в котором размеченный текст включает два параллельных слоя информации: видимая (собственно текст) и скрытая (разметка). При этом скрытая часть информации помещается внутрь текста, но заключается в специальные маркеры <...>, которые отделяют ее от видимого текста. В отличие от внешних способов аннотирования (например, комментариев), разметка всегда инкорпорирована в текст и составляет его неотъемлемую часть. Важным достоинством SGML/XML по сравнению с другими языками разметки (TeX, RTF) является строгий синтаксис команд разметки, различение элементов и атрибутов, явное указание границ элемента, самодокументированность, возможность автоматической проверки правильности.

Среди существующих стандартов кодирования корпусной информации наиболее авторитетными являются: TEI (Text Encoding Initiative) [1], XCES (XML Corpus Encoding Standard) [2], EAGLES (European Advisory Group on Language Engineering Standards) [3]. TEI как наиболее детально разработанный стандарт определяет правила представления практически любых текстов и элементов

текстовой информации, включая: структуру, заголовки, типы речи (проза, поэзия, драма), страницы, цитаты, сноски, ссылки, исправления, таблицы, формулы, специальные символы, лингвистические аннотации, и др. Специальная глава стандарта посвящена правилам кодирования корпусов. Хотя TEI специально не ориентирован на корпусные приложения, он и его производные применяются в большинстве существующих корпусов, включая British National Corpus (BNC), Чешский национальный корпус, Венгерский национальный корпус и другие. Стандарт XCES, являющийся развитием TEI, предназначен только для корпусных применений и определяет множество специфичных для корпусов тегов, однако менее распространен в реальных корпусах.

Однако при детальном рассмотрении универсальные стандарты типа TEI или XCES оказались слишком сложными, избыточными и неудобными для массовой разметки текстов. Полные правила TEI очень обширны и не всегда мотивированы, поэтому соблюсти все требования стандарта достаточно трудно. Формат не отличается компактностью, поэтому часто разметка разрастается без увеличения содержательной информации. Формат нарушает принцип прозрачности, например, мета-атрибуты предлагается записывать как текст внутри тегов, поэтому после удаления разметки исходный текст восстанавливается неточно.

В принципе, можно ограничиться небольшим подмножеством TEI и игнорировать все «лишние» теги. Для представления корпуса можно выбрать следующий минимальный набор тегов из TEI Guidelines: <text> — текст, <p> — абзац, <s> — предложение, <w> — слово, а морфологический разбор записывать в атрибуте <w ana=...>. Однако в таком урезанном виде разметка корпуса вряд ли может претендовать на соответствие стандарту. Скорее она напоминает упрощенный вариант HTML с несколько другим набором тегов.

Но все же главной проблемой для указанных XML-ориентированных форматов является не их сложность, а практическое отсутствие общедоступных программ для подготовки, обработки, индексирования и поиска. В распоряжении лингвистов имеются лишь относительно простые программы: XML-анализаторы, редакторы, конверторы, программы линейного поиска. Понятно, что для корпуса объемом в миллионы слов этого набора совершенно недостаточно. Конечно, внутренние проблемы подготовки и раз-

метки корпуса можно решить при помощи специально написанных конверторов, макросов и других средств. Однако для размещения корпуса в публичный доступ (Интернет) необходима мощная поисковая система, которая умеет эффективно индексировать и быстро находить в корпусе релевантную информацию среди миллионов словоупотреблений, закодированных в выбранном XML-формате.

Сформулируем основные требования к поисковой системе:

1) поиск слов и словосочетаний по их признакам (грамматическим, семантическим и др.);

2) учет контекста и расстояния между словами;

3) поиск метатекстовой информации;

4) развитый язык запросов, включающий логические связки, скобки и контекстные операторы;

5) эффективность индексирования;

6) высокая скорость ответа на любой самый сложный запрос;

7) масштабируемость до очень больших размеров (сотни миллионов словоупотреблений).

Среди доступных программных средств на роль поисковой системы для корпуса подходят следующие классы программ:

1) реляционные базы данных;

2) XML-ориентированные базы данных;

3) полнотекстовые поисковые системы.

Самым сложной проблемой для большинства поисковых систем является **контекстный** поиск. Многие поисковые системы сносно работают на небольших массивах, но при увеличении объема корпуса начинают буксовать. При этом скорость поиска может зависеть не только от объема корпуса, но и от сложности запроса, причем иногда не линейно, а экспоненциально в зависимости от числа слов, участвующих в запросе.

Реализация контекстного поиска требует огромных объемов памяти, потому что приходится хранить точную позицию каждого слова (и его признаков) в многомиллионном массиве текстов. Позиция слова обычно задается в виде тройки: номер документа — номер предложения — номер слова в предложении. При реализации такой системы в реляционной базе данных получаются таблицы, где в каждой строке хранится позиция слова (в виде тройки чисел) и набор атрибутов слова (лексема и ее признаки). Для кон-

текстного поиска сочетания из двух слов требуется операция соединения (join) как минимум двух таблиц, каждая из которых содержит миллионы строк. На самом деле в реляционных системах данные обычно хранятся в более фрагментированном виде, расфасованном по множеству таблиц. Например, для экономии места лексемы и признаки обычно выносятся в отдельные таблицы, а в основной таблице хранятся только ссылки на них. Тогда при любом сложном контекстном запросе система должна выполнять соединение нескольких десятков таблиц огромного объема. Именно поэтому реляционные системы нередко дают нелинейный рост времени ответа при увеличении сложности запроса.

Теоретически для корпуса текстов в формате XML должны идеально подойти XML-ориентированные базы данных (Berkley DB XML, Exist, Ozone, Tamino, Xindice). Однако они появились совсем недавно и не достигли зрелости и эффективности реляционных систем. Как правило, они ориентированы на работу с большим количеством относительно небольших документов. Следуя логике этих систем, нам пришлось бы выделить каждое предложение в отдельный документ, что абсолютно неприемлемо. Кроме того, оказалось, что контекстный поиск невозможно сформулировать на языке запросов к XML-структуре, даже если пронумеровать все слова и предложения. С другой стороны, средства поиска по структуре оказываются не очень нужными, если использовать простую модель текста (слово, предложение, абзац). Получается, что сильные стороны XML-систем фактически не используются, а их слабые стороны не позволяют решить насущные задачи корпуса.

Для эффективной работы с корпусом необходима полнотекстовая поисковая система, специально оптимизированная для контекстного поиска в огромном массиве текстов. Наиболее известные системы такого рода — поисковые интернет-серверы Яндекс и Google, а также другие системы, ориентированные на индексирование и поиск интернет-ресурсов. Для русского корпуса естественным выбором оказался Яндекс-сервер, который обладает очень высокой эффективностью и масштабируемостью, полностью реализует контекстный поиск, включает мощный язык запросов, допускает гибкую настройку индексатора для нестандартных типов текстов, хорошо поддерживает русский язык. Яндекс-сервер выполняет сложные запросы в огромном массиве текстов корпуса за доли секунды, причем объем корпуса никак не влияет на скорость поиска.

2.2. Формат разметки текстов в НКРЯ.

Формат представления информации в корпусе был разработан с учетом существующих стандартов кодирования для корпусов (TEI, XCES), но в качестве ориентира был выбран язык HTML, который является фактическим стандартом для представления текстов в Интернет. TEI и XCES — сложные и экзотические форматы, у которых довольно слабая программная поддержка. HTML является самым распространенным форматом из семьи SGML/XML и поддерживается множеством программ. Поисковые системы научились понимать структуру и семантику тегов HTML, но не могут понять семантику произвольного XML-документа, а могут лишь формально проиндексировать его структуру.

HTML — очень простой формат, который предъявляет минимальные требования к объему и содержанию разметки, а для практической работы можно использовать небольшое подмножество команд. Формат очень компактен и удобен для визуального восприятия и ручного редактирования. Правда, в стандарте нет тегов для представления языковых единиц, но поскольку HTML допускает использование нестандартных тегов, то эта проблема решается путем специальной настройки поискового сервера.

Формат корпуса представляет собой подмножество HTML, к которому добавлено несколько специфических тегов для лингвистических единиц. Формат определяет правила кодирования всей существенной информации о тексте, включая:

1) метатекстовые атрибуты;

2) структурные элементы текста (заголовки, абзацы, стихи, сноски, таблицы);

3) лингвистические единицы (предложения, слова, группы);

4) лексическую информацию (грамматические, семантические признаки);

5) параметры оформления текста, специальные символы и т. д.

Формат был частично адаптирован к требованиям и возможностям Яндекс-сервера, но в целом не сильно отличается от существующего стандарта. Файл в корпусе всегда имеет правильную HTML-структуру, включая стандартный заголовок HTML и собственно текст. Метатекстовые атрибуты естественно обозначаются при помощи стандартных тегов <meta>, которые позволяют представить произвольный набор атрибутов, а не закрытое множество, зафик-

сированное в TEI. Заголовок может содержать другую служебную информацию, например, сведения об источнике и происхождении текста.

Основной текст представляется как последовательность абзацев различных типов, среди которых можно выделить следующие: обычный текст, заголовок, стихотворный фрагмент (строфа), эпиграф, сноска, реплика и другие. Тип абзаца естественным образом кодируется в атрибуте class: ... , таким образом, в данной модели не нужно изобретать специальные теги для обозначения разных структурных единиц текста. В минимальном варианте разметки тип абзаца может не обозначаться, тогда вся разметка сводится к выделению границ абзацев.

В начале текста должен находиться заголовок в формате «Имя Фамилия. Название», заключенный в теги ... . Заголовки разделов документа заключаются в теги ... , где *число* обозначает иерархический уровень элемента: 1 = часть, 2 = глава, 3 = раздел внутри главы. Уровень 1 используется только для частей больших произведений, иначе заголовки верхнего уровня получают номер 2. Так размечаются только явные заголовки (содержащие слова «часть», «глава» или одиночный номер), а немаркированные разделы размечаются как простые абзацы. Текст в заголовках должен быть записан с правильной капитализацией, а не всеми прописными, например: «Глава 1. Рассвет», а не «ГЛАВА 1. РАССВЕТ». Большие фрагменты текста, выделенные прописными, следует переводить в правильный регистр, чтобы их правильно обрабатывал парсер; отдельные слова, выделенные прописными, можно оставить как есть.

Стихотворные фрагменты оформляются как абзацы со стилем ... , границы строк внутри стихов обозначаются через
.

Сноска воспроизводится как абзац со стилем ... и размещается непосредственно после абзаца, откуда на нее идет ссылка, либо все сноски помещаются в конец файла. Рекомендуется оформлять номер сноски тегом ^{...}, а также связать ссылку на сноску и саму сноску гипертекстовой ссылкой ... и .. .

Таблицы воспроизводятся как набор абзацев без сохранения табличного расположения. В простых случаях каждая строка таблицы воспроизводится как абзац, а ячейки разделяются табуляцией. Для сложных таблиц, где ячейка может занимать несколько строк, каждая ячейка воспроизводится как отдельный абзац. Необходимо, чтобы ячейка содержала связный текст, а не куски, вырванные из разных столбцов.

Шрифтовое оформление отдельных фрагментов текста воспроизводится как есть, без попыток его семантической интерпретации. Используются стандартные теги HTML: <b> (жирность), <i> (курсив), (разрядка), <sup> (верхний индекс), <sub> (нижний индекс).

Символы, отсутствующие в русской кодировке Windows, записываются при помощи стандартных кодов HTML типа é или ā. Знаки ударения в русских словах ставятся после буквы и записываются как ́ (акэт) и ̀ (гравис). Латинские буквы с диакритиками также записываются при помощи кодов HTML или упрощенно без диакритик.

Простые формулы типа a^2+bi_2 могут записываться как текст с нужным оформлением (курсив, жирность, индексы), а сложные формулы удаляются.

Для обозначения лингвистических единиц используются нестандартные теги, частично заимствованные из TEI: <w> — слово, <st> — предложение (вместо <s>, который в HTML занят для обозначения зачеркивания). Лексическая информация помещается в атрибуты тега <w> и имеет следующий вид:

<w lex='*лексема***' gr='***грамматические признаки***' sem='***семантические признаки***'>**слово**</w>**

Грамматические и семантические признаки внутри атрибутов gr и sem разделяются пробелами, запятыми и другими небуквенными символами. Варианты морфологического разбора разделяются символом «;», причем порядок перечисления вариантов в атрибутах lex и gr совпадает. Порядок признаков и конкретный разделитель важны для человека, читающего разметку, а для поисковой машины это безразлично, поскольку все признаки попадают в индекс на равных основаниях.

Предложенный формат кодирования лексической информации близок к оптимальному и отвечает всем требованиям, сформули-

рованным в начале п. 2. Этот формат достаточно компактен, полон и легко расширяем, поскольку при необходимости всегда можно добавить новые атрибуты. Формат полностью прозрачен, поскольку исходный текст сохраняется без изменения, а вся лингвистическая информация скрыта в атрибутах. Формат согласован с правилами HTML/XML и легко стыкуется с большим числом программ, включая поисковый индексатор, морфологический парсер, разнообразные конверторы и редакторы. Все это открывает широкие возможности для автоматизации разметки в корпусе.

3. Технология подготовки текстов для корпуса.

Процесс подготовки текстов для корпуса включает следующие основные этапы:

1) предварительная разметка текста в минимальном HTML-формате;

2) морфологическая разметка и снятие омонимии (в части корпуса);

3) метатекстовая разметка;

4) преобразование в выходной формат для Яндекс-сервера.

На каждом последующем этапе по сравнению с предыдущим объем и содержательность дополнительной информации все время увеличивается. На первом этапе в текст вносится информация о его формальной структуре, размечаются различные типы элементов текста, параметры оформления, специальные символы. На втором этапе в текст добавляется собственно лингвистическая (морфологическая) информация. На третьем этапе появляется «паспорт» текста в виде набора метатекстовых атрибутов, который готовится отдельно от самого текста. На конечном этапе метатекстовая информация объединяется с текстом, который, пройдя еще ряд трансформаций, загружается и индексируется Яндекс-сервером. Лишь после этого текст становится частью корпуса и доступным для поиска.

Для поддержки технологического процесса разработан комплекс программ и методик, позволяющих автоматизировать наиболее трудоемкие операции на различных этапах подготовки текста. Понятно, что некоторые операции (снятие омонимии, метаразметка) в принципе невозможно сделать автоматически, но можно создать комфортную среду для их выполнения в автоматизированном режиме под контролем человека.

Возможность автоматизации в значительной степени зависит от согласованности форматов представления текста на различных этапах обработки. Именно это позволяет связать все операции в единую технологическую цепочку, так что выходные данные одной программы становятся входными данными для следующей, и так далее до конечного результата. Эта связка становится возможной потому, что на всех этапах технологии используются текстовые форматы: простой текст, HTML с различной глубиной разметки, текст с разделителями и другие. Текстовые форматы являются самыми простыми, понятными, универсальными и могут обрабатываться множеством программ, в отличие от сложных и закрытых двоичных форматов (DOC, XLS, PDF).

В технологии широко используется язык программирования Perl, который имеет мощные средства для обработки текстов: глобальный поиск и замена с применением регулярных выражений (шаблонов замен) и динамические структуры данных (ассоциативные массивы). На нем написано большинство сервисных программ (конверторы, фильтры, программы проверки), так что Perl играет роль «клея», который связывает все компоненты в единую цепочку.

3.1. Предварительная разметка.

Тексты для корпуса поступают из разных источников и бывают представлены в самых разнообразных форматах: простой текст, HTML, RTF, PDF, и другие. Для каждого входного формата создается набор конверторов и макросов, позволяющих преобразовать исходный текст в нужный вид — HTML с минимальной структурной разметкой, описанный в п. 2.2. В конверторах и макросах широко используются расширенные возможности Perl и Winword: глобальный поиск и замена с использованием регулярных выражений.

Рассмотрим наиболее распространенные исходные форматы:

1) Текст без оформления или с простейшим оформлением (plain text).

В этом формате абзацы обычно обозначаются при помощи начальных пробелов, часто сохраняются переносы слов, шрифтовые выделения обозначаются символами _ * и т. д. При переводе в HTML лишние пробелы и переносы убираются, абзацы заменяются на теги <p>, размечаются структурные элементы (заголовки, стихи), добавляются шрифтовые команды, создается правильный

заголовок файла. Для таких текстов разработан набор конверторов и макросов (глобальных замен), позволяющих быстро получить минимальную HTML-разметку, которая потом вручную доводится до требуемого вида.

2) Текст с «богатой» HTML-разметкой, полученный из Интернета.

Основная проблема таких текстов — огромный объем избыточной разметки, не относящейся к содержанию текста, а служащей исключительно для дизайна. Сюда относятся всевозможные баннеры, счетчики, скрипты, навигационные ссылки, меню, таблицы, иконки, которые повторяются на каждой странице и нередко занимают от 50 до 90% ее объема. Весь этот мусор порой совершенно затмевает содержание страницы и требуются огромные усилия, чтобы выделить содержание из окружающего его художественного оформления. Почти для каждого сайта необходимо разрабатывать специальный фильтр, который выделяет содержание текста, но даже после фильтра приходится вручную исправлять и удалять избыточную разметку.

3) Формат Winword (RTF).

Для этого формата разработан набор макросов и замен, позволяющих получить HTML с достаточно богатой разметкой, которая сохраняет большую часть оформления оригинала: шрифтовые выделения, специальные символы, различные стили абзацев, таблицы и др. Дело в том, что существующие конверторы из Winword в HTML порождают чрезмерно сложный и избыточный код, который очень трудно очистить и привести к нужному виду. Поэтому оказывается проще извлечь необходимый минимум разметки из оформления оригинала, находясь внутри Winword и используя его продвинутые возможности. Таким образом, Winword используется в двух функциях: как система подготовки документов с богатым оформлением, и как простой текстовый редактор для глобальных замен.

4) PDF и форматы издательских систем.

Эти форматы вообще могут быть обработаны только после конвертирования в один из стандартных форматов: RTF (Winword) или простой текст. Все зависит от наличия и возможностей существуюших конверторов. Большинство программ позволяет сохранить документ в одном из указанных форматов, после чего он обрабатывается по технологии Winword или как простой текст.

5) Табличные данные в текстовом формате с разделителями (CSV).

Некоторые новостные данные были представлены в виде таблицы, которая содержит метатекстовые атрибуты и собственно текст (обычно очень короткий). Для таких данных был разработан конвертор, который из каждой строки таблицы генерирует файл HTML, содержащий текст и метатекстовые атрибуты в нужном формате.

При подготовке текста из него удаляются все элементы, не являющиеся авторским текстом или несущественные для изучения языка, в частности:

- колонтитулы, номера страниц;
- титульные страницы, содержание, выходные данные, бибописание, аннотация;
- редакторские сноски (авторские сноски сохраняются);
- рисунки, схемы, формулы (но подписи под ними сохраняются);
- длинные последовательности чисел (в таблицах).

Если исходный файл представляет собой сборник, он разделяется на отдельные тексты в соответствии с оглавлением. Предисловия и комментарии (кроме авторских) оформляются как отдельные тексты, если они представляют интерес для корпуса, иначе выбрасываются.

Многие элементы оформления оригинала представляются в упрощенном виде, например, таблицы не сохраняются в табличном виде, стили абзацев различаются лишь при необходимости, шрифтовое оформление заголовков не сохраняется.

Несмотря на большой набор сервисных программ, предварительная разметка остается весьма трудоемкой операцией, в которой много работы приходится выполнять вручную, приспосабливаясь к особенностям конкретного текста. Именно на первом этапе должен быть достигнут определенный уровень качества текста, без которого текст невозможно передать на дальнейшую обработку и поместить в корпус.

3.2. Морфологическая разметка.

Морфологический парсер — это программа, которая приписывает каждому слову текста все возможные грамматические разборы, иногда с учетом синтаксического контекста. Парсер обычно включает в себя словарь лексем и правила словоизменения, а так-

же правила предсказания для анализа незнакомых форм, отсутствующих в словаре.

В корпусе используются два парсера:

1) Dialing (компания «АОТ») для корпуса со снятой омонимией;

2) Mystem (компания «Яндекс») для корпуса с неснятой омонимией.

Dialing на самом деле является мощным синтаксическим анализатором для русского языка, который строит синтаксическую структуру предложения и при этом ему удается снять часть морфологической омонимии. Mystem — это чисто морфологический анализатор, который всегда выдает все варианты без учета синтаксического контекста. Mystem имеет мощный механизм предсказания разборов для незнакомых форм, а Dialing в этом случае нередко ошибается.

Парсер представляет собой очень сложную программу, которую трудно модифицировать, а попытки улучшить его работу могут привести к непредсказуемым результатам. Вместо того, чтобы исправлять работающую программу, проще написать программный фильтр, который будет исправлять ошибки первой программы.

Грамматический фильтр исправляет результат работы парсера, в частности:

- удаляет ненужные варианты разбора, присутствующие в словаре (например, несклоняемые существительные «ли», «он»);
- помечает варианты, маловероятные в данном контексте (например, предложный падеж после предлога «к»);
- переупорядочивает и заменяет грамматические признаки; и др.

Грамматические фильтры написаны на языке Perl отдельно для каждого парсера: Gram.bat для Mystem и Dial.bat для Dialing. В обоих случаях после фильтра получается текст с более чистой разметкой и меньшим числом ошибок, которые пришлось бы исправлять вручную.

Выходной формат парсера отличается от конечного формата корпуса, а именно: морфологический разбор ставится после словоформы и заключается в фигурные скобки, внутри скобок варианты разделяются знаком “|”. Этот промежуточный формат выбран специально для ручного снятия омонимии, поскольку он менее

громоздок и в нем лучше видна разница между разметкой HTML (<...>) и морфологической разметкой ({...}).

Для ручного снятия омонимии разработан специальный редактор (Gramedit), который представляет собой не автономную программу, а надстройку над Winword. Таким образом разметчик попадает в знакомую среду и может использовать все возможности этого мощного редактора. Для визуального выделения разные элементы текста оформляются разными цветами и стилями, в частности:

- варианты разбора и команды разметки оформляются как скрытый текст и в обычном режиме не видны;
- словоформы оформляются разными цветами в зависимости от числа вариантов разбора: ноль, один или несколько.

Варианты разбора для текущей словоформы выдаются в виде списка, из которого нужно выбрать правильный вариант или отредактировать существующий. Редактор позволяет перемещаться по тексту, делать глобальные замены и отменять исправления.

Основные проблемы при ручном снятии омонимии таковы:

1) Форма неизвестна парсеру. В этом случае парсер порождает несколько гипотетических вариантов, из которых нужно выбрать наиболее близкий к правильному и исправить его вручную. Здесь очень легко ошибиться в порядке и правильном написании граммем.

2) Форма неизвестна парсеру и является омонимом другой известной формы, например, фамилия «Грибов» разбирается как род. множ. от «гриб». В этом случае правильный вариант также приходится писать вручную или копировать из другого похожего варианта.

3) Слово записано с орфографическим искажением (дэвушка, да-а-а-а-й). В этом случае нужно найти в тексте правильный вариант разбора и скопировать его.

4) Выдается несколько почти одинаковых вариантов, отличающихся мелкими деталями (разговаривать=V,несов=... и разговаривать=V=несов,...) и без семантического комментария. В таких случаях для различения омонимов приходится обращаться к грамматическому словарю, чтобы понять, как смысловое различие связано с грамматическим.

В результате работы должен получиться текст со снятой омонимией, где каждому слову приписан ровно один вариант разбора.

Результирующий документ должен быть сохранен в текстовом формате, чтобы его можно было передать на следующие этапы обработки.

3.3. Метатекстовая разметка.

Метатекстовые атрибуты могут приписываться текстам независимо от обработки самих текстов, поэтому этапы 2 и 3 могут выполняться параллельно и в произвольном порядке. Необходимо только, чтобы текст был идентифицирован и имел фиксированное имя файла. После этой фиксации никакие слияния, разделения и переименования файлов недопустимы, так как они разрушили бы всю систему.

Для хранения метаинформации используются обычные таблицы Excel заранее заданной структуры, где в первом столбце находится имя файла (с путем), а в остальных столбцах метатекстовые атрибуты и другая технологическая информация. Это позволяет пользоваться встроенным средствами Excel для поиска, фильтрации, анализа и проверки данных (списки значений, автозаполнение, статистика). При этом таблицы должны храниться в текстовом формате с разделителями (табуляциями), который понимает Excel. Это позволяет включить таблицы в обработку при помощи внешних программ, также понимающих этот формат.

Теоретически метаинформация могла бы всегда храниться отдельно от самих текстов, однако по правилам HTML она должна находиться в заголовке файла, чтобы Яндекс-сервер мог ее проиндексировать. При раздельном хранении метаинформации возникает постоянная проблема синхронизации и согласованности между мета-таблицами и текстами. Для решения этой проблемы разработан следующий комплекс программ:

1) Программа Metas собирает метатекстовые атрибуты из заголовков файлов и создает заготовку мета-таблицы, которая дальше правится вручную в среде Excel. Дело в том, что уже на первоначальном этапе обработки в тексты может быть внесена некоторая метаинформация, например, автор, заголовок и дата создания. На заключительном этапе программа Metas.bat может опять собрать атрибуты для их окончательной проверки.

2) Программа Meta2txt переносит метатекстовые атрибуты из исправленных мета-таблиц в реальные тексты. Она проверяет наличие файла и обновляет заголовок. Множественные значе-

ния атрибута в таблицах разделяются символом "|", а при переносе в текст каждое значение превращается в отдельный атрибут. Таким образом, метатекстовые атрибуты могут свободно перемещаться между текстами и мета-таблицами в обоих направлениях, а метаразметка может выполняться итеративно с несколькими циклами проверки.

3) Программа MetaTest проверяет правильность мета-таблицы путем сранения значений атрибутов с нормативной таблицей, содержащей правильные значения или их шаблоны. Программа помечает неверные значения символом "#", чтобы их можно было проверить и исправить вручную.

Все указанные выше программы реализованы на языке Perl.

После окончательной проверки метатекстовая информация объединяется с размеченным текстом и получается законченная и самоценная информационная единица, которая может быть загружена в Интернет или использоваться автономно для различных научных задач.

3.4. Преобразование в выходной формат.

На заключительном этапе обработки подготовленные тексты с метаразметкой и морфологической разметкой проходят еще несколько автоматических трансформаций. При этом используются следующие программы (написанные на языке Perl):

1) Конвертор, преобразующий рабочий формат разметки в окончательный.

Конвертор переводит морфологические разборы в фигурных скобках в правильный формат <w lex=... gr=...>, проверяет некоторые ошибки разметки, переводит имена признаков в латиницу, добавляет недостающие признаки (действительный залог, полная форма), добавляет видовые пары глаголов для повышения качества поиска.

2) Программа семантической разметки (Semmarkup).

Программа приписывает словам основные семантические признаки из специального семантического словаря. Это позволяет осуществить семантический поиск в части корпуса со снятой омонимией. Семантический словарь оформлен в виде таблицы, в первых столбцах которой записана лексема и часть речи, а в остальных находятся семантические признаки. Программа сравнивает морфологические признаки слова со словарем и при совпадении пере-

носит семантические метки в атрибут sem тега <w>. В случае многозначных слов это создает некоторый шум при семантическом поиске, но пока снятия семантической многозначности не предполагается.

3) Статистические программы (Gramstat, Metastat).

Эти программы собирают статистику распределения для грамматических и метатекстовых признаков в текстах. Это позволяет обнаружить ошибки в значениях признаков до окончательной загрузки. Программа Gramstat позволяет получать распределение по отдельным частям морфологического разбора (лексема, часть речи, грамматические признаки лексемы и словоформы) и их комбинациям.

Описанная выше технология помогает автоматизировать значительную часть рутинных операций при подготовке текстов для корпуса. Некоторые операции остаются принципиально неавтоматизируемыми (чистка текстов, снятие омонимии, метаразметка), но для них также создан набор сервисных средств для облегчения работы. Формат кодирования информации на начальных этапах работы специально сделан очень простым, чтобы лишняя разметка не мешала воспринимать и редактировать текст при ручной обработке. Выходной формат со сложной разметкой генерируется автоматически на окончательном этапе.

Литература

[1] Guidelines for Electronic Text Encoding and Interchange. *Ed. by C. M. Sperberg-McQueen and Lou Burnard.* http://www.tei-c.org/P4X/index.html.

[2] Corpus Encoding Standard for XML. http://www.cs.vassar.edu/XCES/.

[3] Recommendations for the Morphosyntactic Annotation of Corpora. EAGTCWG-MAC/R. http://www.ilc.pi.cnr.it/EAGLES96/browse.html