УДК 519.7.004

А.А. ПЛУГИН, Л.Э. ЧАЛАЯ

МЕТОД АВТОМАТИЧЕСКОГО ГЕНЕРИРОВАНИЯ БИБЛИОГРАФИЧЕСКИХ СПИСКОВ

Описывается разработанный автором алгоритм и программная реализация web-сервиса для автоматической генерации библиографических списков. Web-сервис разрабатывается с использованием языка программирования Perl и технологии ASP.NET. Графический интерфейс пользователя реализуется с помощью технологии ASP.NET. Интерфейс пользователя представляет собой веб-приложение, применять которое можно через веб-браузер.

Введение

В современном обществе существует большое количество информации – книг, журналов, статей, диссертаций и т.п., представленных в различных видах – электронном, бумажном. Это обилие порождает трудности в поиске информации, преодолеваемые путем различных систематизаций. Для систематизации литературных произведений используют библиографические описания литературных источников, которые должны однозначно их идентифицировать. Проблема, возникающая при составлении и корректировке библиографических описаний и списков, – сложность и трудоемкость учета всех требований действующих стандартов оформления. Решению этой проблемы может способствовать разработка web-сервиса для генерации библиографических списков в соответствии со стандартом государства, в котором выполняется научная или другая работа, требующая включения в себя таких списков.

1. Постановка задач исследования

Цель работы – создание web-сервиса для автоматической корректировки списков библиографических описаний. Web-сервис должен предоставлять пользователю интерфейс взаимодействия с системой автоматической корректировки и формирований библиографических описаний.

Для достижения цели необходимо решить следующие задачи: исследовать стандарты и правила оформления библиографических списков; проанализировать методы обработки текстовой информации; исследовать применимость регулярных выражений для обработки библиографических описаний; обосновать выбор программных средств для реализации web-сервиса; разработать алгоритм автоматического формирования библиографических списков; разработать программный web-сервис, позволяющий использовать систему автоматической корректировки библиографических списков.

2. Анализ предметной области, регулярных выражений и формальных грамматик

Библиографическое описание – совокупность библиографических сведений о документе, его составной части или группе документов, приведенных по определенным правилам и необходимых и достаточных для общей характеристики и идентификации документа. Источником библиографических сведений является документ в целом, в первую очередь те его элементы, которые содержат выходные сведения [1]. Библиографические сведения в описании показывают в том виде, в каком они даны в документе, или формулируют их на основе анализа документа. Набор и последовательность использования источников библиографических сведений регламентированы в соответствующих принятых стандартах. Для уточнения имеющихся или получения недостающих библиографических сведений применяют библиографические пособия, библиотечные каталоги, справочные издания и прочие источники. Библиографические сведения, заимствованные из них, а также сведения, сформулированные на основе анализа документа, заключают в квадратные скобки.

Для каждой области библиографического описания установлен основной источник библиографических сведений. При отсутствии такого источника его заменяют другим, содержащим наиболее полную информацию. Библиографические сведения, заимствованные не из основного источника, могут быть заключены в квадратные скобки. Библиографическое описание состоит из элементов, объединенных в области, и заголовка. Элементы и области приводят в последовательности, установленной в перечнях стандарта. Отдельные элементы и области могут повторяться. Библиографические сведения, относящиеся к разным элементам, но грамматически связанные в одном предложении, записывают в предшествующем элементе. Элементы библиографического описания подразделяют на обязательные и факультативные. Обязательные элементы обеспечивают идентификацию документа. Их приводят в любом библиографическом описании при наличии соответствующих сведений в источнике описания. Общий для группы библиографических описаний обязательный элемент, вынесенный в название раздела или в заглавие информационного издания, в описании опускают. Факультативные элементы дают дополнительную информацию о документе (его содержании, читательском назначении, иллюстративном материале и т.п.). Набор факультативных элементов определяет библиографирующее учреждение. Он является постоянным для определенного информационно-поискового массива (библиотечного каталога, издания и т.д.).

В библиографическом описании областям и элементам предшествуют следующие условные разделительные знаки: «.—» (точка и тире), «.» (точка), «,» (запятая), «:» (двоеточие), «;» (точка с запятой), «/» (косая черта), «//» (две косые черты), «( )» (круглые скобки), «[ ]» (квадратные скобки), «+» (плюс), «=» (знак равенства). Каждой области библиографического описания, кроме первой, предшествует знак точка и тире (. —), который заменяют точкой, если область выделена шрифтом или записана с новой строки. Если первый элемент области отсутствует, точку и тире ставят перед последующим элементом, разделительный знак которого в этом случае опускают. При повторении в библиографическом описании отдельных областей (серии, примечания, международного стандартного номера) повторяют точку и тире.

Библиографическое описание составляют на языке текста документа, при этом часть сведений (например, в области количественной характеристики) допускается записывать на языке того государства, в котором находится библиографирующее учреждение. Библиографическое описание может быть составлено на языке выходных сведений, если язык текста и язык выходных сведений различны. При необходимости библиографическое описание составляют в транслитерации, транскрипции, а также в переводе на язык того

государства, в котором находится библиографирующее учреждение. Во всех случаях, когда язык библиографического описания отличается от языка текста документа или его выходных сведений, в области примечания приводят данные о языке текста или выходных сведений документа.

В библиографическом описании следует соблюдать нормы современной орфографии. Первое слово каждого элемента (кроме сведений об иллюстрациях) начинают с прописной буквы. Остальные прописные буквы приводят в соответствии с нормами языка, на котором составлено описание, независимо от того, какие буквы приведены в документе. При наличии в документе ошибок и опечаток, не искажающих смысла, сведения в описании приводят в исправленном виде и не оговаривают исправления. Ошибки и опечатки, изменяющие смысл текста, а также все ошибки в фамилиях, инициалах лиц, принимавших участие в создании документа, в датах, исправляют. Сведения об ошибочной форме приводят в области примечания или отмечают иным способом.

При составлении библиографического описания применяют различные приемы сокращений. В библиографических описаниях документов, изданных на русском и других языках, для часто встречающихся в различных областях и элементах библиографического описания слов и понятий применяют унифицированные формы сокращений на русском языке: «и др.» (и другие), «и т.д.» (и так далее), «ст.» (старший), «мл.» (младший), «Б. м.» (Без места), «Б. и.» (Без издательства), «Б. г.» (Без года), «Разд. паг.» (Раздельная пагинация).

В библиографических описаниях документов на иностранных языках с латинской графической основой для приведенных слов и понятий применяют унифицированные формы сокращений на латинском языке: “etc.” (et cetera), “et al.” (et alii), “Sen.” (Senior), “Jun.” (Junior), “S. I.” (Sine loco), “S. n.” (Sine nomine), “S. a.” (Sine anno), “pag. var.” (pagine varia). При необходимости их эквиваленты приводят на соответствующих языках. В отдельных случаях, например, при длинном заглавии, опускают часть элемента или фразы, при этом пропуск обозначают знаком многоточие (…).

Существует такое понятие, как выражение в квадратных скобках. Квадратные скобки ограничивают поиск теми символами, которые в них заключены – «[abc]». Этому регулярному выражению соответствует любая строка, содержащая abc либо вместе, либо каждый из них в отдельности. Предположим, что необходимо создать регулярное выражение, соответствующее всем буквам русского алфавита. В этом случае можно перечислить все эти буквы в регулярном выражении, это допустимо. Более коротко такое регулярное выражение можно записать как «[а-Я]». Оно соответствует всем буквам русского алфавита, поскольку любые два символа, разделяемые дефисом, задают соответствие диапазону символов, находящихся между ними. Регулярное выражение «[а-Я]» описывает символы как нижнего, так и верхнего регистров, поэтому более подробно это выражение можно записать как «[а-яА-Я]». Точно таким же образом задаются регулярные выражения, соответствующие числам как «[0-9]» или как «[0123456789]». Оба этих выражения эквивалентны и соответствуют любой цифре.

Языком в алфавите V называется подмножество цепочек конечной длины, состоящих из символов этого алфавита. Для описания языков существует несколько способов, однако в данной работе интерес представляет только один – с помощью порождающих грамматик. Порождающая грамматика G – это четверка (VT, VN, P, S), где: VT – алфавит терминальных символов (или терминалов); VN – алфавит нетерминальных символов (или нетерминалов), который не пересекается с VT; P – конечное множество правил вывода; S – нетерминальный ($S \in VN$) начальный символ грамматики.

Языком, порождаемым грамматикой G = (VT, VN, P, S), называется множество всех цепочек терминальных символов, выводимых из начального символа в данной грамматике. Грамматики G_1 и G_2 называются эквивалентными, если порождаемые ими языки одинаковы. Цепочка б, выводимая из S в грамматике G = (VT, VN, P, S), называется сентенциальной формой в грамматике G. Таким образом, язык, порождаемый грамматикой G, также можно определить как множество терминальных сентенциальных форм [5].

Ноам Хомский в своих работах выделил четыре основных типа грамматик, в зависимости от их условной сложности. Чем меньше «номер» типа, тем грамматика считается

более сложной. Для отнесения грамматики к тому или иному типу необходимо соответствие всех ее правил вывода некоторым схемам [9]. Самая общая грамматика – типа 0, которая называется неограниченной грамматикой. К неограниченным грамматикам (или грамматикам с фазовой структурой) относятся все формальные грамматики без исключения.

Формальная грамматика $G = (VT, VN, P, S)$ называется неограниченной или грамматикой типа 0, если все ее правила имеют вид $\alpha \to \beta$, где $\alpha \in (VT \cup VN)^+$ содержит хотя бы один нетерминальный символ, $\beta \in (VT \cup VN)$. В силу своей сложности этот тип грамматик представляет исключительно теоретический интерес, на практике они не применяются. К типу 0 обычно относят естественные языки.

К типу 1 относят контекстно-зависимые и неукорачивающие грамматики. Неукорачивающей называется формальная грамматика $G = (VT, VN, P, S)$, все правила вывода которой имеют вид $\alpha \to \beta$, где $\alpha \in (VT \cup VN)^+$, $\beta \in (VT \cup VN)$ и $(\alpha) \le (\beta)$ (т.е. каждая последующая цепочка вывода этой грамматики будет иметь такую же или большую длину). Контекстно-зависимой грамматикой называется формальная грамматика $G = (VT, VN, P, S)$, все правила которой имеют вид $\lambda_1 A \lambda_2 \to \lambda_1 \beta \lambda_2$, где $\lambda_1, \lambda_2 \in (VT \cup VN)$ $A \in VN$, $\beta \in (VT \cup VN)$. Множество языков, порождаемых неукорачивающими грамматиками, совпадает со множеством языков, порождаемым контекстно-зависимыми грамматиками, поэтому все грамматики типа 1 можно называть контекстно-зависимыми. Контекстно-зависимые грамматики используются для анализа текста на естественных языках, но в силу своей сложности при трансляции они не применяются.

Контекстно-свободной называется формальная грамматика $G = (VT, VN, P, S)$, все правила вывода которой имеют вид $A \to \beta$, $A \in VN$, $\beta \in (VT \cup VN)$. Укорачивающие контекстно-свободные грамматики почти эквивалентны контекстно-свободным. Практически для всех языков программирования можно построить порождающие их контекстно-свободные грамматики. Именно поэтому стандартный способ описания синтаксиса языков программирования, данных, протоколов (в документах RFC и ISO) – форма Бэкуса-Наура является своеобразной формой контекстно-свободных грамматик.

Формальные грамматики типа 3 называются регулярными. Регулярные грамматики бывают двух типов: праволинейной и леволинейной. Праволинейные и леволинейные грамматики эквивалентны. Праволинейной называется формальная грамматика $G = (VT, VN, P, S)$, все правила вывода которой имеют вид $A \to tB$, где $A, B \in VN$, $t \in VT$.

Леволинейной называется формальная грамматика $G = (VT, VN, P, S)$, все правила вывода которой имеют вид $A \to Bt$, где $A, B \in VN$, $t \in VT$.

Регулярные языки (т.е. языки, порождаемые регулярными грамматиками) могут быть описаны и другим способом – с помощью так называемых регулярных выражений [3]. Можно определить понятие регулярного выражения рекурсивно:

а) ε – регулярное выражение, порождающее пустой регулярный язык $\{\varepsilon\}$;

б) a – регулярное выражение, порождающее регулярный язык {a};

в) пусть P и Q – регулярные языки, которые порождены регулярными выражениями p и q соответственно. Тогда:

1) (p|q) – регулярное выражение, которое порождает регулярный язык P|Q ;

2) (pq) – регулярное выражение, которое порождает регулярный язык PQ;

3) (p*) – регулярное выражение, которое порождает регулярный язык P*;

4) других регулярных выражений нет.

Истоки регулярных выражений лежат в теории автоматов, теории формальных языков и классификации формальных грамматик – по Хомскому. Эти области изучают вычислительные модели (автоматы) и способы описания и классификации формальных языков. В 1940-х гг. Уоррен Маккалок и Уолтер Питтс описали нервную систему, используя простой автомат в качестве модели нейрона [8]. Математик Стивен Клини позже описал эти

модели, используя свою систему математических обозначений, названную «регулярные множества». Кен Томпсон встроил их в редактор QED, а затем в редактор ed под UNIX. С этого времени регулярные выражения стали широко использоваться в UNIX и UNIX-подобных утилитах, таких как expr, awk, Emacs, vi, lex и Perl.

В регулярных выражениях лишние скобки обычно опускаются, при этом действует соглашение о приоритете операций: наивысшим приоритетом обладает операция итерации, затем идет операция конкатенации и, наконец, операция объединения имеет самый низкий приоритет. Регулярные выражения называются эквивалентными, если они порождают один и тот же язык. Отношение эквивалентности будем обозначать знаком равенства (=). Сейчас регулярные выражения используются многими текстовыми редакторами и утилитами для поиска и изменения текста на основе выбранных правил. Многие языки программирования уже поддерживают регулярные выражения для работы со строками. Например, Perl и Tcl имеют встроенный в их синтаксис механизм обработки регулярных выражений. Набор утилит (включая редактор sed и фильтр grep), поставляемых в дистрибутивах Unix, одним из первых способствовал популяризации понятия регулярных выражений.

Регулярные выражения используются для сжатого описания некоторого множества строк с помощью шаблонов, без необходимости перечисления всех элементов этого множества. При составлении шаблонов применяется специальный синтаксис, поддерживающий следующие операции:

– перечисление, вертикальная черта разделяет допустимые варианты. Например, «gray|grey» соответствует gray или grey;

– группировка, круглые скобки, используются для определения области действия и приоритета операторов. Например, «gray|grey» и «gr(a|e)y» являются разными образцами, но они оба описывают множество, содержащее gray и grey;

– квантификация, фигурные скобки, находится после символа или группы и определяет, сколько раз предшествующее выражение может встречаться; {m,n} определяет общее выражение, повторений может быть от m до n включительно; {m,} определяет общее выражение, m и более повторений; {,n} определяет общее выражение, не более n повторений. Знак вопроса означает 0 или 1 раз, то же самое, что и {0,1}, например, «colou?r» соответствует и color, и colour. Звёздочка означает 0, 1 или любое число раз {0,}. Например, «go*gle» соответствует gogle, ggle, gooogle и т. д. Плюс означает хотя бы 1 раз {1,}, например, «go+gle» соответствует gogle, google и т. д. (но не ggle).

Существует большое количество вариаций синтаксиса регулярных выражений, зависящего от реализации. Синтаксис «базовых» регулярных выражений Unix на данный момент определён POSIX как устаревший, но он до сих пор широко распространён из соображений обратной совместимости. Многие Unix-утилиты используют этот синтаксис по умолчанию. В синтаксисе регулярных выражений Unix большинство символов соответствуют сами себе («a» соответствует «a» и т. д.). Исключения из этого правила называются метасимволами:

– . (точка) соответствует любому единичному символу;

– [ ] соответствует любому единичному символу из числа заключённых в скобки.

Символ «-» интерпретируется буквально только в том случае, если он расположен непосредственно после открывающей или перед закрывающей скобкой: [abc-] или [-abc]. В противном случае он обозначает интервал символов. Например, [abc] соответствует «a», «b» или «c»; [a-z] соответствует буквам нижнего регистра латинского алфавита. Эти обозначения могут и сочетаться: [abcq-z] соответствует a, b, c, q, r, s, t, u, v, w, x, y, z. Чтобы установить соответствие символам «[» или «]», достаточно, чтобы закрывающая скобка была первым символом после открывающей: [][ab] соответствует «]», «[», «a» или «b»;

– [^] соответствует единичному символу из числа тех, которых нет в скобках. Например, [^abc] соответствует любому символу, кроме «a», «b» или «c»; [^a-z] соответствует любому символу, кроме символов нижнего регистра в латинском алфавите;

– ^ соответствует началу текста или началу любой строки;

– $ соответствует концу текста или концу любой строки;

– \ объявляет «отмеченное подвыражение», которое может быть использовано позже (см. следующий элемент: \n). «Отмеченное подвыражение» также является «блоком»;

– \n где n – это цифра от 1 до 9; соответствует n-му отмеченному подвыражению. Эта конструкция теоретически нерегулярна, она не была принята в расширенном синтаксисе регулярных выражений;

– * после выражения, соответствующего единичному символу, соответствует нулю или более копий этого выражения. Например, «[xyz]*» соответствует пустой строке, «x», «y», «zx», «zyx» и т. д. Комбинация символов \n*, где n – это цифра от 1 до 9, соответствует нулю или более вхождений для соответствия n-го отмеченного подвыражения. Например, «\(a.\)c\1*» соответствует «abcab» и «abcaba», но не «abcac». Выражение, заключённое в «\(» и «\)» и сопровождаемое «*», следует считать неправильным. В некоторых случаях оно соответствует нулю или более вхождений строки, которая была заключена в скобки. В других оно соответствует выражению, заключённому в скобки, учитывая символ «*»;

– \{x,y\} соответствует последнему блоку, встречающемуся не менее x и не более y раз. Например, «a\{3,5\}» соответствует «aaa», «aaaa» или «aaaaa».

Различные реализации регулярных выражений интерпретируют обратную косую черту перед метасимволами по-разному. Например, egrep и Perl интерпретируют скобки и вертикальную черту как метасимволы, если перед ними нет обратной косой черты, и воспринимают их как обычные символы, если черта есть.

Многие диапазоны символов зависят от выбранных настроек локализации. В POSIX было стандартизовано объявление некоторых классов и категорий символов (таблица).

Классы и категории символов

	POSIX	Unix-синтаксис	Perl	Обозначение
Класс либо категория	[:upper:]	[A-Z]		символы верхнего регистра
	[:lower:]	[a-z]		символы нижнего регистра
	[:alpha:]	[A-Za-z]		символы верхнего и нижнего регистра
	[:alnum:]	[A-Za-z0-9]		цифры, символы верхнего и нижнего регистра
	–	[A-Za-z0-9_]	\w	цифры, символы верхнего, нижнего регистра и « »
	–	[^A-Za-z0-9_]	\W	не цифры, символы верхнего, нижнего регистра и «_«
	[:digit:]	[0-9]	\d	цифры
	–	[^0-9]	\D	не цифры
	[:xdigit:]	[0-9A-Fa-f]		шестнадцатеричные цифры
	[:punct:]	[.,!?:…]		знаки пунктуации
	[:blank:]	[\t]		пробел и TAB
	[:space:]	[\t\n\r\f\v]	\s	символы пробелов(пропуска)
		[^ \t\n\r\f\v]	\S	не символы пробелов(пропуска)
	[:cntrl:]	[\x00-\x1F\x7F]		символы управления
	[:graph:]	[:alnum:] ? [:punct:]		символы печати
	[:print:]	[\x20-\x7E]		символы печати и символы пропуска (видимые символы и пробелы)

Способ представления самих метасимволов «.», «,», «-», «[», «\» и других в регулярных выражениях без интерпретации, т.е., в качестве простых (не специальных) символов – экранирование их обратной косой чертой: «\». Например, чтобы представить сам символ «точка», необходимо написать «\.». Чтобы представить символ открывающей квадратной скобки «[», надо написать «\[» и т.д. Сам метасимвол \ тоже может быть экранирован, т.е. представлен как две обратных косых черты, и тогда интерпретатор регулярных выражений воспримет его как простой символ обратной косой черты.

Квантификаторам в регулярных выражениях соответствует максимально длинная строка из возможных (квантификаторы являются «жадными»). Это может оказаться значи-

тельной проблемой. Например, часто ожидают, что выражение (<.*>) найдёт в тексте теги HTML. Однако этому выражению так же соответствует любая строка, заключенная между двумя тегами. Проблему жадности квантификаторов можно решить двумя способами. Первый состоит в том, что в регулярном выражении учитываются символы, не соответствующие желаемому образцу (<[^>]*> для описанного выше случая). Второй заключается в определении квантификатора как нежадного (ленивого) – большинство реализаций позволяют это сделать, добавив после него знак вопроса. Например, выражению (<.*?>) будет соответствовать только тег HTML [2].

Следовательно, существуют «ленивые» аналоги «жадных» квантификаторов:

*? – «не жадный» («ленивый») эквивалент *;

*? – «не жадный» («ленивый») эквивалент *;

+? – «не жадный» («ленивый») эквивалент +;

{n,}? – «не жадный» («ленивый») эквивалент {n,}.

Использование «ленивых» квантификаторов может повлечь за собой обратную проблему, когда выражению соответствует слишком короткая строка.

Также существуют квантификаторы повышения жадности. Сверхжадные квантификаторы (possessive quantifiers):

*+ – «сверхжадный» эквивалент *;

++ – «сверхжадный» эквивалент +;

{n,}+ – «сверхжадный» эквивалент {n,}.

Регулярные выражения в POSIX аналогичны традиционному Unix-синтаксису, но с добавлением некоторых метасимволов:

«+» указывает на то, что предыдущий символ или группа может повторяться один или несколько раз. В отличие от звёздочки, хотя бы одно повторение обязательно;

«+» указывает на то, что предыдущий символ или группа может повторяться один или несколько раз. В отличие от звёздочки, хотя бы одно повторение обязательно;

«|» разделяет альтернативные варианты регулярных выражений. Один символ задаёт две альтернативы, но их может быть и больше, достаточно использовать больше вертикальных чёрточек. Необходимо помнить, что этот оператор применяет максимально возможную часть выражения. По этой причине оператор альтернативы чаще всего используется внутри скобок.

Регулярные выражения в Perl имеют более богатый и в то же время предсказуемый синтаксис, чем в POSIX. По этой причине очень многие приложения применяют именно Perl-совместимый синтаксис регулярных выражений. В нём используются такие группы:

( ) простая группа с захватом;

(?:) группа без захвата. Аналогична группе с захватом, но заключённое в скобках выражение не добавляется к списку захваченных фрагментов. Например, если требуется найти или «здравствуйте», или «здрасте», но не важно, какое именно приветствие найдено, можно воспользоваться выражением здра(?:сте|вствуйте);

(?=) группа с заглядыванием вперёд. Продолжает поиск только если справа от текущей позиции в тексте находится заключённое в скобки выражение. При этом само выражение не захватывается. Например, говор(?=ить) найдёт «говор» в «говорить», но не в «говорит».

3. Метод генерации и его программная реализация

В работе предложен метод решения поставленной задачи. Схема его алгоритма представлена на рис. 1.

Алгоритм работы программы заключается в четырёх основных этапах:

а) пользователь вводит в программу библиографическое описание, которое может быть оформлено некорректно;

б) программа осуществляет разбор полученного библиографического описания на атомарные информационные единицы;

в) если возможно (информация достаточно полна), то осуществляется форматирование полученных данных в соответствии со стандартами;

г) для пользователя выводятся сформированные конечные и промежуточные данные.

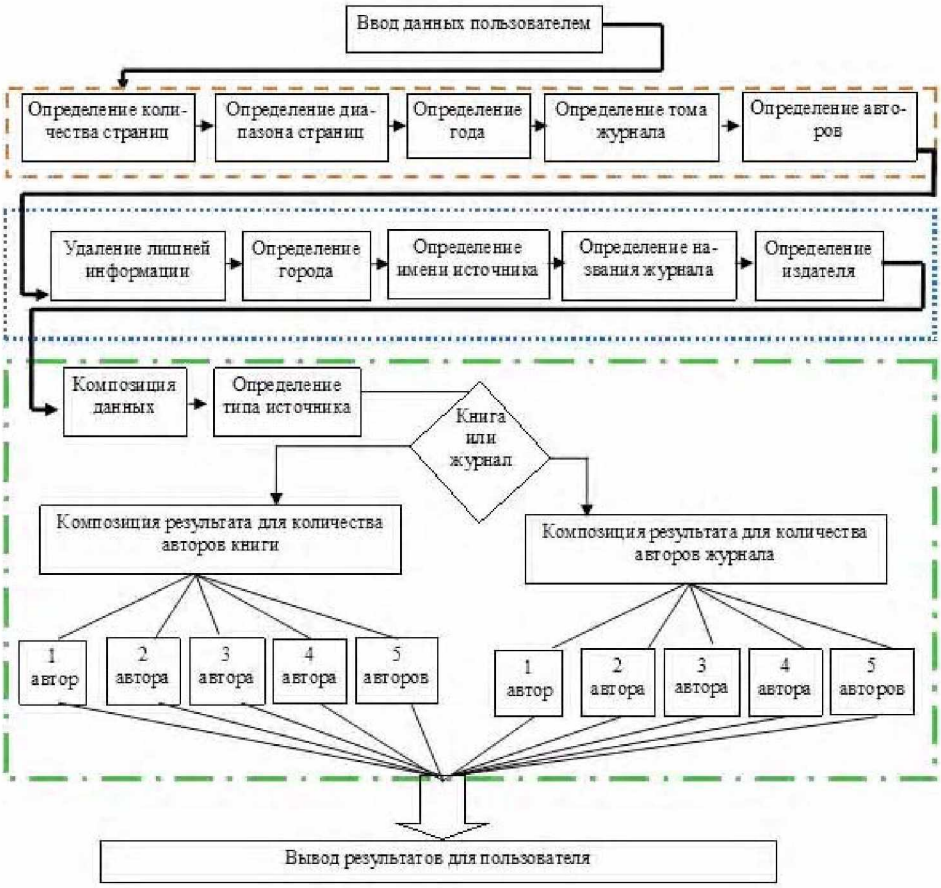

Рис. 1. Схема алгоритма метода генерации

Первый и последний этапы касаются периферийных возможностей ввода и вывода программы и не влияют на результат работы. Второй этап – самая объёмная и трудоёмкая часть программной реализации данной дипломной работы. На втором этапе происходит разбиение полученного описания на следующие информационные единицы: тип литературного источника; название литературного источника; название издательства; город издательства; количество страниц; диапазон страниц; список авторов; название периодического издания; номер периодического издания; том периодического издания.

Перечисленные информационные единицы находятся в глобальной области видимости, т.е. являются общими для всех типов литературных источников, но так как некоторые информационные единицы уникальны, то, в идеальном случае, невозможно наличие сразу всех информационных единиц во входящем библиографическом описании. На деле такая ситуация может существовать по причине того, что пользователь не ограничен рамками на оформление входящего библиографического описания. Однако чем больше степень соответствия входящего библиографического описания стандарту, тем больше вероятность того, что операция разбиения на информационные единицы завершится успешно.

Процесс разбиения заключается в трёх подэтапах. Первый подэтап – это попытка извлечь из библиографического описания однозначно-определённую информацию, т.е. информацию, структура которой всегда находится в определённых рамках. Например, год – это всегда четырёхзначное число, чаще всего с последующей точкой после этого числа. Количество страниц – это число, с последующей буквой «С», которая может быть малой либо большой, после нее часто стоит точка. Данные утверждения не гарантируют абсолютное отсутствие проблем, связанных с ложными совпадениями. Так, может встретиться ситуация, что в названии книги будет присутствовать конструкция, аналогичная описанной выше. Однако стоит заметить, что такие ложные совпадения маловероятны, либо специфические – для литературы определённого рода. Извлечение однозначно определённой ин-

формации реализовано с помощью регулярных выражений, которые могут однозначно описывать определённые конструкции. Всего на этом этапе возможно извлечь следующие информационные единицы: количество страниц; диапазон страниц; список авторов; номер периодического издания; том периодического издания. Второй подэтап заключается в удалении из входящего библиографического описания уже найденной информации, а также вырезании лишних несущественных символов, таких как точка или двойные пробелы.

На третьем подэтапе происходит поиск информационной единицы «город издательства» методом перебора всех возможных городов, где могут располагаться издательства. Этот способ поиска города достаточно непроизводителен, на сравнение строки с каждым возможным городом тратится большое количество вычислительной мощности, однако другого возможного решения найдено не было. Обычно название литературного источника располагается слева от города, а название издательства (либо название периодического издания) – справа. Поэтому, используя город как разделитель строки, можно извлечь следующие информационные единицы: название литературного источника; название издательства; название периодического издания. Стоит заметить, что у периодического издания не должно быть города, поэтому разделителем строки там служит конструкция «//», которая является обязательной для описания периодических изданий. Соответственно, невозможно корректно выделить эти информационные единицы, если не указана конструкция «//». В примере приведена глобальная функция извлечения всех информационных единиц:

```
sub parseSource($)
{
my ( $source ) = @_;
$pagesCount = findPages($source);
$pageDiapason = findPageDiapason($source);
$year = findYear($source);
my $magazineNumberArrayRef = findMagazineNumber($source);
$magazineNumberStruct = $magazineNumberArrayRef->[0];
$magazineNumber = $magazineNumberArrayRef->[1];
my $magazineVolumeArrayRef = findMagazineVolume($source);
$magazineVolumeStruct = $magazineVolumeArrayRef->[0];
$magazineVolume = $magazineVolumeArrayRef->[1];
@authors = findAuthors($source);
my $cutedSource = prepareToPositionBasedCutting($source);
my @cities = loadCities();
$city = findCity(\@cities, $cutedSource);
$name = findName($cutedSource);
$magazineName = findMagazineName($cutedSource);
$publisher = findPublisher($cutedSource);
$composedString = Composer::Compose(
$pagesCount, $pageDiapason, $year,
$city, $name, $publisher, $magazineName,
$magazineNumber, $magazineVolume, @authors);
saveResult($source);
}
```

В конце этой функции происходит вызов модуля «Composer», который составляет из полученных информационных единиц корректный литературный источник, т.е. реализует третий этап работы программы.

Третий этап заключается в компоновке полученных результатов в готовое библиографическое описание. На этом этапе важно знать тип литературного источника, так как в зависимости от того, книга это либо периодическое издание, описание будет формироваться различными способами. Заключение о типе литературного источника делается на основании нахождения информационных единиц «количество страниц» и «диапазон страниц». Различные выводы делаются по причине того, что в библиографическом описании книги не должно быть диапазона страниц, а в описании статьи периодического издания этот параметр должен быть, но должна отсутствовать информационная единица «количество страниц».

На базе алгоритма было создано программное приложение, которое генерирует корректное библиографическое описание. Для реализации приложения был использован язык программирования Perl и технология ASP.NET. Интерфейс пользователя представляет собой веб-приложение, пользоваться которым можно через веб-браузер. Программа выводит скомпонованное библиографическое описание на web-форму, пример вывода представлен на рис. 2. Web-форма, реализованная на языке C#, отображает вывод, полученный самой программой, написанной на языке Perl, используя промежуточный файл.

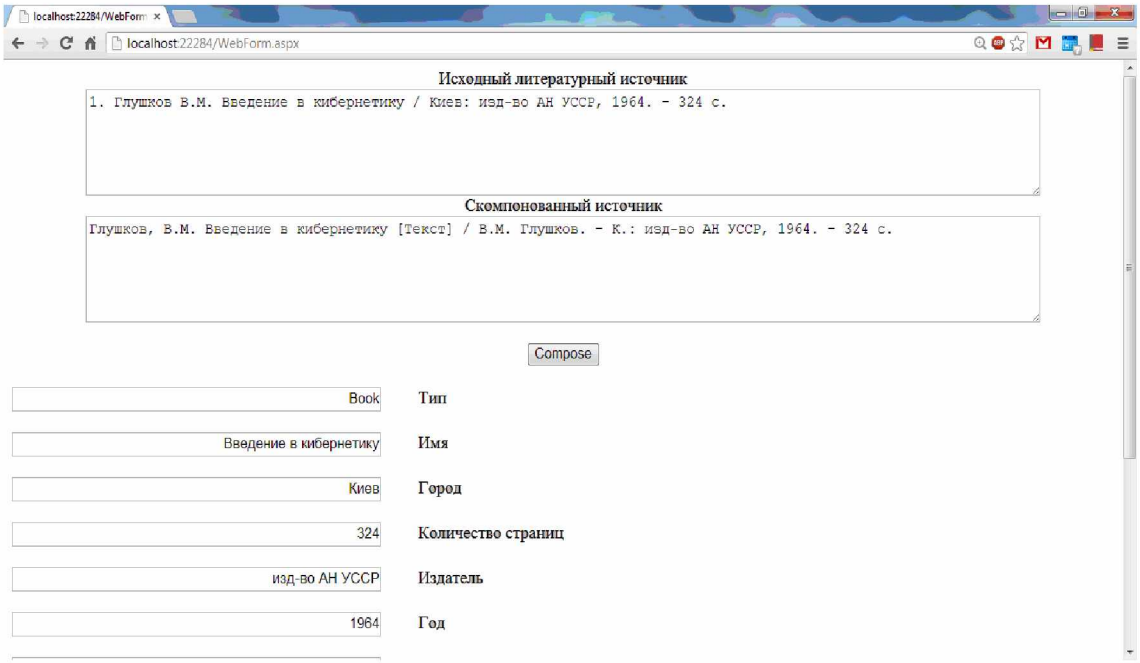

Рис. 2. Внешний вид web-приложения и вывод данных в web-форму

В качестве примера работы программы приведены скриншоты web-форм с результатами преобразования библиографических описаний с некорректной формой в стандартизированные. На рис. 3 приведен пример преобразования книги с авторством 3-х человек.

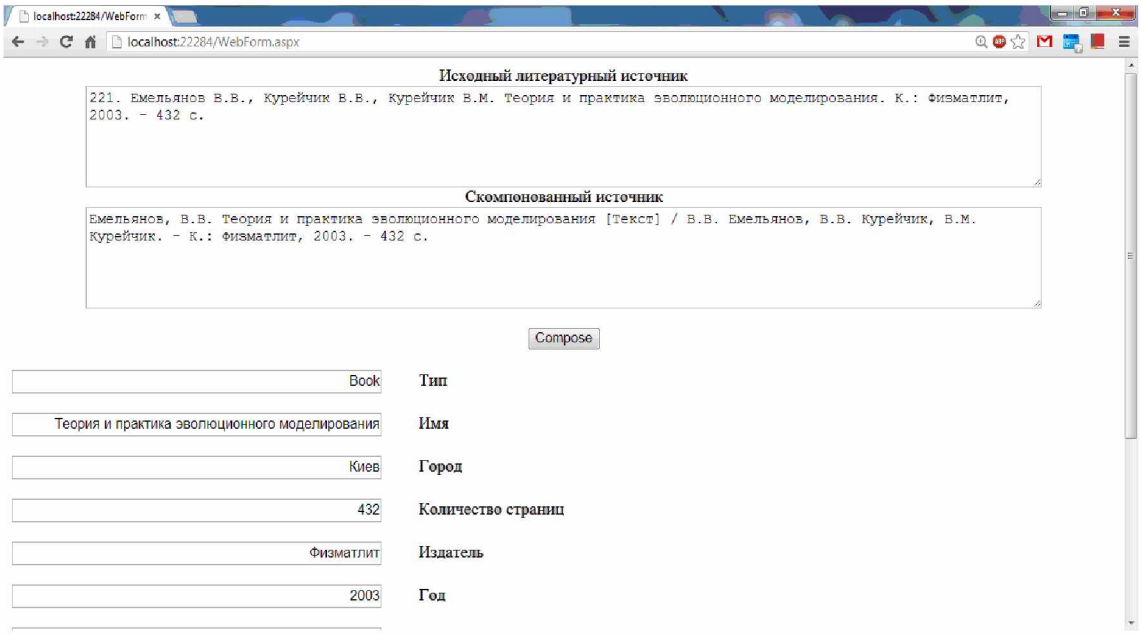

Рис. 3. Преобразование описания книги с тремя авторами

Преобразование библиографических описаний, ссылающихся на периодические издания, приводит к другому результату. Пример такого преобразования статьи авторства 4-х человек приведен на рис. 4. Видно, что алгоритм одинаково работает со ссылками на источники как на русском языке, так и на украинском.

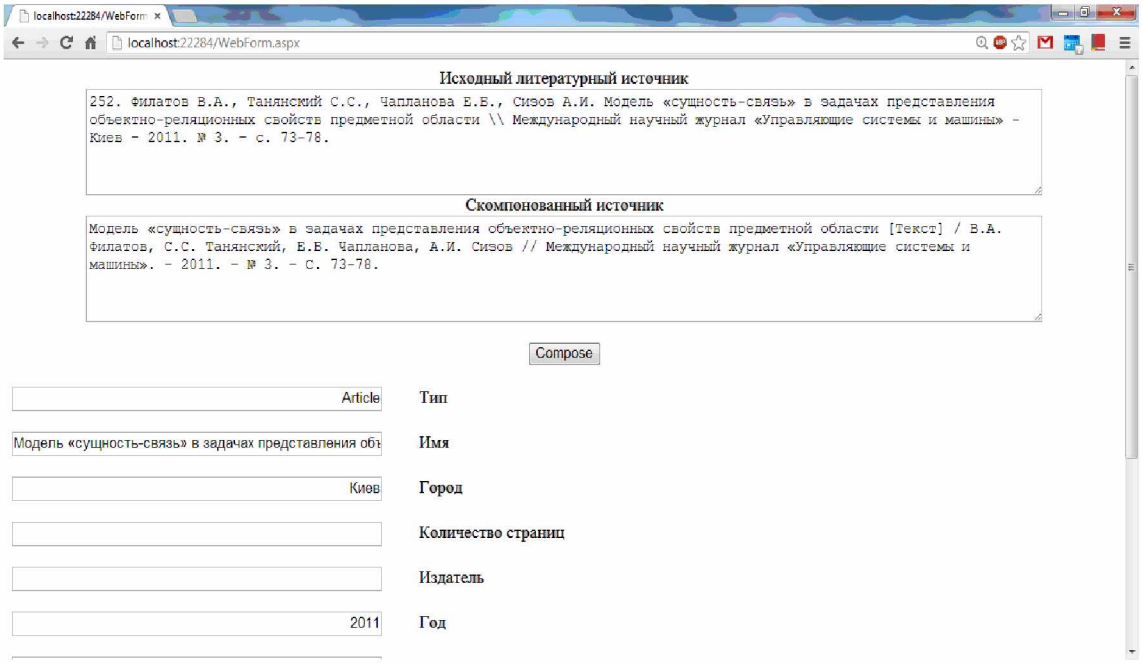

Рис. 4. Преобразование описания статьи с четырьмя авторами

Программа имеет недостатки, среди которых отсутствие поддержки множества языков. Поддержка ограничивается русским и украинскими языками. Это ограничение вызвано тем, что некоторые однозначно определённые информационные единицы могут иметь различные формы на различных языках. Методом решения этого недостатка является усовершенствование регулярных выражений, а также реализация дополнительных регулярных выражений, учитывающих языковые особенности. Трудности возникают также с определением информационной единицы «город» – для поддержки конкретного языка необходимо вводить дополнительную базу данных с перечнем всех городов, в которых есть издательства, причём для каждого поддерживаемого языка.

Другой недостаток связан с невозможностью сформировать корректное библиографическое описание при недостатке обязательных информационных единиц. Такая ситуация возможна в случае, если пользователь не указал всю информацию, либо когда программа не смогла определить необходимую информацию. Методом решения данной проблемы является интеграция приложения с поисковым роботом, который будет искать недостающую информацию в сети Интернет. В результате анализа существующих сервисов для поиска литературы был сделан вывод о том, что задача реализации поискового робота, который будет взаимодействовать с существующими сервисами, является достаточно сложной, поэтому не рассматривается в данной работе. Более простая задача – извлечение недостающей информации из базы данных библиографических описаний, но не было найдено ни одной такой существующей базы данных.

Ещё один недостаток – это невозможность определения всех информационных единиц. Причина кроется в больших вариациях возможных форм библиографических описаний. По этой же причине алгоритм может определить и обработать только два типа литературных источников – периодические издания и книги. Могут возникать неоднозначности при разборе входящей информации, например, при наличии фамилии и инициалов человека, в названии книги. В таком случае алгоритм определит человека, имя которого указано в названии книги, как одного из авторов.

Таким образом, программа имеет допустимые ошибки и не может обрабатывать некоторое количество библиографических описаний. Эти ошибки устранимы, однако это трудоёмкая работа.

Выводы

1. Предложен метод, который решает актуальную на данный момент проблему оформления библиографических описаний.

2. Актуальность данной работы обоснована тем, что составление библиографических описаний в соответствии с существующими стандартами оформления вызывает затруднения у многих соискателей в научной среде. Даже у людей с большим опытом составления описаний эта задача может занимать достаточно много времени. Разработанное в виде web-сервиса приложение предназначено для ускорения этого процесса.

3. Разработанная программа решает задачу автоматизации процесса формирования библиографических описаний согласно стандартам. Входными данными программы являются библиографические описания источников, которые не соответствуют стандартам, причём степень возможного несоответствия не должна быть слишком большой. Подразумевается, что люди, составляющие эти описания, приблизительно знакомы с общими правилами оформления. Выходными данными являются оформленные в соответствии со стандартами библиографические описания.

Список литературы: **1.** *Гречихин А.А.* Общая библиография: Учебник. [Текст] / А.А. Гречихин. М.: МГУП, 2000. 588 с. **2.** *Мельников С.В.* Perl для профессиональных программистов. Регулярные выражения [Текст] / С.В. Мельников. М.: Бином, 2007. 190 с. **3.** *Смит Б.* Методы и алгоритмы вычислений на строках (regexp) = Computing Patterns in Strings [Текст] / Б. Смит. М.: Вильямс, 2006. 496 с. **4.** *Гойвертс Я.* Регулярные выражения. Сборник рецептов. [Текст] / Я. Гойвертс, С. Левитан. СПб: Символ-Плюс, 2010. 608 с. **5.** *Ахо А.* Теория синтаксического анализа, перевода и компиляции. Синтаксический анализ [Текст] / А. Ахо, Дж. Ульман. М.: Мир, 1978. 524 с. **6.** *Уолл Л.* Программирование на Perl [Текст] / Л. Уолл, Т. Кристиансен, Дж. Орвант. М.: O'Reilly, Символ, 2008. 1145 с. **7.** *Уайтхэд П.* Perl: наглядный курс программирования [Текст] / П. Уайтхэд. М.: Диалектика, 2001. 280 с. **8.** *Mcculoch W.S.* A logical calculus of the ideas immanent in nervous activity [Text] / W.S. Mcculoch, W. Pitt // Mathematical biophysics. 1943. № 5. P. 115-132. **9.** *Хопкрофт Дж.* Контекстно-свободные грамматики и языки [Текст] / Дж. Хопкрофт, Р. Мотвани, Д. Ульман. М.: «Вильямс», 2002. 528 с.