References

1. Makhmudov M.: Systems of automatic recycling of Turkic text on lexical and morphological level, Elm, 114 p. Baku (1991) (In Russian)
2. Migalkin V.V.: Modeling of the Yakut language spelling and development a set of programs to check the spelling of Yakut texts in Windows environment, Author. diss.... Ph.D., Yakutsk (2005) (In Russian)
3. Sadyqov T.: Problems of modeling of Turkic morphology: an aspect of causing Kyrgyz nominal inflectional forms, 119 p. Publishing House of the "Ilim" (1987) (In Russian)
4. Sirazitdinov Z.A.: Modeling grammar of Bashkir language. Inflectional system. 160 p. Ufa (2006) (In Russian)
5. Sirazitdinov Z.A.: On the modeling of inflectional system agglutinative language pair combinations (for example, the Bashkir language) / Actual problems of modern Mongolian and Altaic. Proceedings of the International Scientific Conference. Elista, 2014. pp 139-143. (In Russian)
6. Altenbek G., Wang Xiao-long: Kazakh Segmentation System of Inflectional Affixes. In: Joint Conference on Chinese Language Processing, pp.183-190 (2010)
7. Zafer H.R., Tilki B., Kurt A., Kara M.: Two-level description of Kazakh morphology. In: Proceedings of the first International Conference on Foreign Language teaching and Applied Linguistics, FLTAL 2011, Sarajevo (May 2011).
8. Sharipbaev A.A.: Intelligent morphological analyzer, based on semantic networks: Conference proceedings Open Semantic Technologies for Intelligent Systems (2012)
9. Bekmanova G.T.: Some approaches to the problems of automatic word changes and morphological analysis in the Kazakh language. In: Bulletin of the East Kazakhstan State Technical University Named by D. Serikbayev, №1, pp. 192-197, Ust-Kamenogorsk (2009) (In Russian)
10. Zhubanov A.H.: Basic principles of formalization of the Kazakh text content, 250 p. Almaty (2002) (In Russian)

УДК 81.374:811.512.161:811.512.154

PARSING AND ANNOTATION OF TURKISH-KYRGYZ DICTIONARY

Kadyr Momunaliev, *Kyrgyz-Turkish Manas University kadyr.momunaliev@gmai.com*

The case study described in this article is the first milestone on the way toward a full featured Text Encoding Initiative P5 annotation standard. The paper outlines parsing and annotating workflow to obtain initial XML-based structure of Turkish-Kyrgyz dictionary. Typography-based parsing techniques are implemented in procedural programming language environment; corresponding workflow charts are presented in form of pseudo code and block schemas. Resulted XML dictionary bases are verified and applied in desktop and web e-dictionary implementations. It is proposed that such kind of explicitly structured data representation is easier to manipulate and use as a basement for further deeper lexicographic annotations.

Keywords: dictionary data, structured data, unstructured data, XML, human-computer, typography, semantics, syntax, parsing

ПАРСИРОВАНИЕ И АННОТИРОВАНИЕ ТУРЕЦКО-КЫРГЫЗСКОГО СЛОВАРЯ

Момуналиев Кадыр Замирович, *Кыргызско-Турецкий Университет «Манас» kadyr.momunaliev@gmai.com*

Данное тематическое исследование представляет собой один из пройденных этапов на пути к достижению полноценного стандарта аннотирования Text Encoding Initiative P5.

Статья освещает методологию парсинга и аннотирования результатом которой является XML структурированные данные турецко-кыргызского словаря. Приемы парсинга основанные на типографических свойствах текста реализованы в процедурной среде прогараммирования; соответствующие алгоритмы представлены в виде блок схем и псевдо кода. Полученные XML структуры были верифицированы и применены в офф-лайн и веб приложениях типа электронного словаря. В качестве технического предложения утверждается, что аннотированные и структурированные данные легче поддаются машинной обработки и представляют собой фундамент для более углубленного лексикографического анализа.

Ключевые слова: словарные данные, структурированные данные, неструктурированные данные, XML, человек-компьютер, типография, семантика, синтаксис, парсинг

1. Introduction

Today big amounts of information accumulated in electronic media or in printed form require universal, i.e. globally accessible and efficient way of representation since the world is becoming a worldwide network of information exchange and business transactions [1]. This information from different domains of human activity needs not only physical infrastructure allowing transmission of bytes, files and documents, but should allow transmission of their semantics, i.e. their meaning. In the coming future of Semantic Web computers are going to "understand" the contents of documents, i.e. navigate, read, access and capture relevant information. Human's part of information processing by browsing and reading is less efficient and thus cannot provide a competitive advantage.

In such a setting new standards of information representation that is purposed for computer access are being sought for. Some new technologies allowing keeping information semantics in computer-readable manner has already appeared (XML and RDF). These types of information provide explicit semantic structure and often called machine-readable or machine-accessible formats.

The trend of data conversion to computer-readable format is not an exception for dictionaries. Many dictionaries are under their way to be converted and others are already done. There is a range of formats to encode dictionary data: binary, relational database etc. But why to choose XML –based formats, such as XDXF [4], StarDict Textual File Format, TEI P5 etc.? The main reason is that they may be used as a cross format representations, i.e. universally interchangeable data containers or structures. Many software agents are able to work with XML, since it has open and predictable structure. Even a human being is able to read and edit such files

They don't require special purpose converters to convert from one format to another, since XML query and transformation languages provide mapping standards between different XML schemas [2].

In our case study we have a formatted text representation of a dictionary that should be analyzed to obtain lexical (or editorial) XML encoded representation reflecting semantic structure of the dictionary. Parsing in this context implies splitting up textual body of a human readable document to explicitly shown lexicographical units in form of XML tree. Factually it is a conversion of human readable data to machine readable format. *Figure 1* demonstrates what data is given as input and what may be gotten as output.

[4] XDXF (XML Dictionary eXchange Format) is a project to unite all existing open dictionaries and provide both users and developers with universal XML-based format, convertible from and to other popular formats like Mova, PtkDic, StarDict. Retrieved from www.en.wikipedia.org

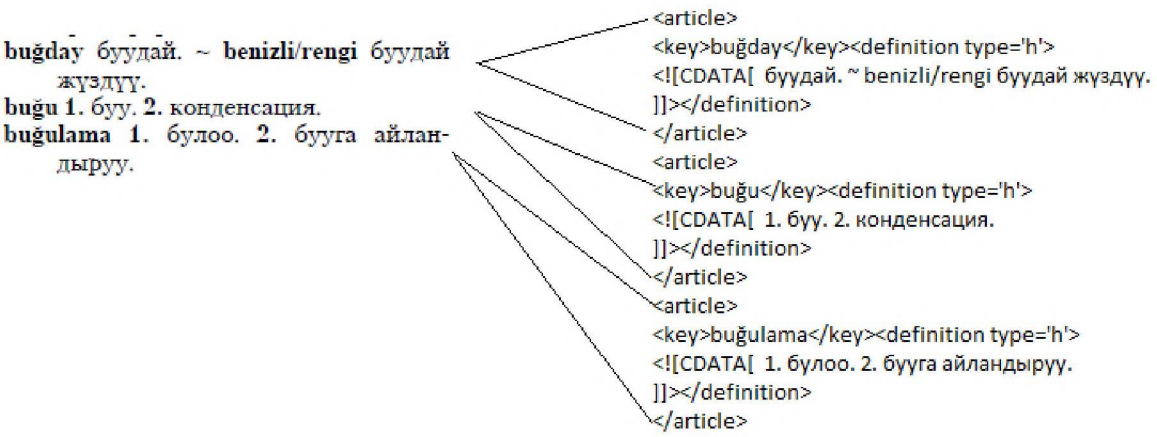


Figure 3. The sample of input(left) and output(right) data (Textual file schema) of the parser.

Why to encode dictionary data? Because if there were no machine-readable formats machines/computers would have to generate parsers for every dictionary they encounter since every dictionary may have its own specific structure and rules to interpret the content. Secondly dictionaries are rather static information sources that may be parsed once for further multiple usages.

2. Input Data

The Turkish-Kyrgyz Dictionary by Gulzura Cumakunova[5] (hereinafter Cumakunova's dictionary) has been chosen as an input data to be parsed and annotated. The dictionary is structured and formatted according to international lexicographical rules. In the input file all headings, footers, page numbers, first matter, last matter has been temporarily removed, and their annotations are not subject of this paper. The input data implies a series of dictionary entries readable by human. The main body of the dictionary data has hierarchical structure and is shown in form of a schematic tree, see *Figure 2*.

Description of *Figure 2:*

1. All dictionary entries belong to the dictionary, i.e. <dictionary/> unit is the root element
2. There must be at least one dictionary entry in the <dictionary/>, i.e. <entries/>1+
3. Each entry consists of mandatory form and definition parts: <form/> and <definition/>. Form part contains Turkish source text and definition part contains Kyrgyz target text.
4. Form block consists of two parts: 1st part contains headword (<headword/>) that may be single headword or the first word of the compound phrase, or acronym; 2nd part contains information related to <headword/> that may be special ending causing headword to inflect (<ending/>), or it may contain the compound phrase(s) in full view, or very close synonym, or acronym's expanded view. (see the tree leaves at the left)
5. Definition block is divided by two areas or blocks: sense and usage (in the tree <sense_block/> and <usage_block/>).

a. Sense block contains one or more sense items which in turn comprise one or more options of translation equivalents very close by meaning and delimited by commas. Translation equivalents may be accompanied by sense example which consists of Turkish source text and Kyrgyz target text. Full stop character delimits translation equivalents from sense example. And finally there may be two kinds of references: 'see' reference and 'compare' reference. 'See' reference comprises all the sense item space and nothing may accompany it. 'Compare' reference is an appendix for translation(s) (as like as sense example). Both of references are preceded by

[5] More information about the dictionary author could be found on: https://ky.wikipedia.org/wiki/Жумакунова,_Гүлзура

reserved abbreviations in Kyrgyz(‘к.’ abbreviated form of ‘кара’ – ‘see’, and ‘сал.’ abbreviated form of ‘салыштыр’ – ‘compare’).

b. Usage block is optional and if it occurs then it may consist of one or more usage items (i.e. samples of stereotypic phrases, idioms, proverbs etc) which don’t necessarily explain the certain sense of the headword, but show its usage. Usage items are not attached to certain sense item, instead they refer to the headword in general. There is no special delimiter dividing sense items group from usage items, but usage items come after sense items. Any usage item is represented by source Turkish phrase and its target Kyrgyz translation(s) delimited by round bracket enumeration (like ‘1)’, ‘2)’ etc). Translations of usage item are not terminated by full stop.

6. Additionally any leaf of the tree (rather sense or usage children) may contain author’s explanation note placed in the round brackets and outlined by italic font. (This is not shown in the diagram.)

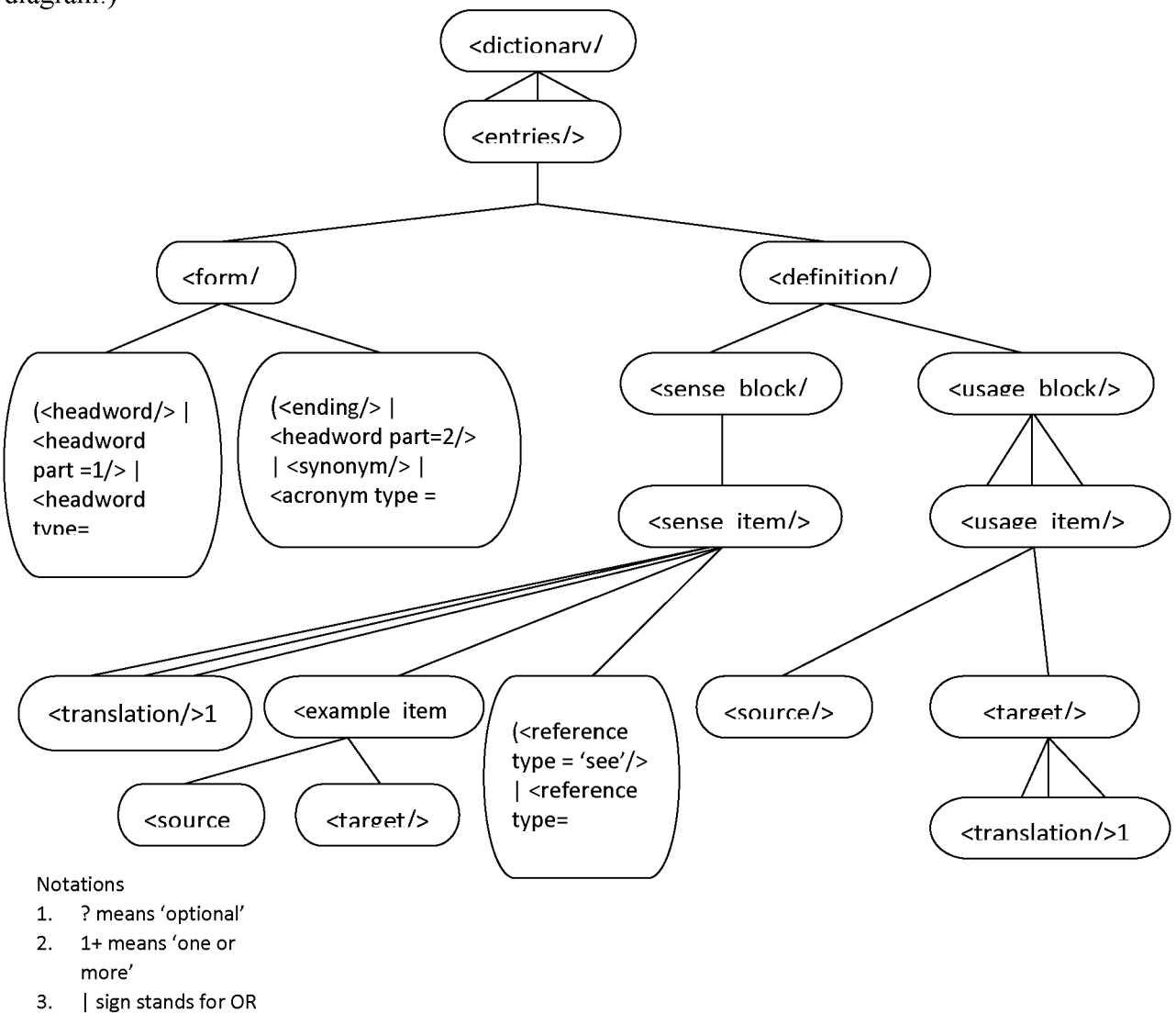


Figure 4. Lexicographical structure of the dictionary data.

The parser is intended to make use of typography features (pre-annotated, i.e. converted to XML tags), syntax and reserved constructions that described in the following tables (*Table 1,2.3*). Dictionary’s typographical features and their corresponding lexicographical meanings are summarized in *Table 1*

Table 1. Typography-to-semantics interpretation schema

	Typographic Indicators			Semantics or Lexicographical meaning
Content unit	Layout	Formatting	Character set	
Text		Bold font	Latin Turkish	Turkish content (Headword, idiom, stereotypic phrases and other source matter)
Text		Normal font	Cyrillic Kyrgyz	Kyrgyz content (senses, translations and other target matter)
Text		Italic font	Latin	International names of flora and fauna species
			Cyrillic Kyrgyz	Additional explanations
			Cyrillic Kyrgyz	abbreviations[6]
Line	Out dented, i.e. hanging, i.e. shifted to the left			First line of the entry
Word	locates at the beginning of hanging line	Bold font	Latin Turkish	Headword
Text	locates at the beginning of a hanging line	Bold font	Latin Turkish	Form block[7]
Word			First letter capitalization	Proper name
Word			Full capitalization	Acronym

Syntax, i.e. punctuation meanings are shown in *Table 2.*

Table 2. Syntax of the dictionary or punctuation semantics

Punctuation name	Notation	Additional indicators	Function or meaning
Swung dash	~	bold	Headword placeholder
Slash	/		Delimits equally suitable words in a given context
Full stop	.		Sense item's end
			Usage item's end
			Example item's end
			Sense block translations' end
			Accompanies abbreviations

[6] Kyrgyz abbreviations are followed by mandatory full stop and don't change, i.e. they are fixed pre-defined words.
[7] There are two exceptions with 'же' (or) and acronym expanded form, see section Entry Parsing

		bold	Accompanies sense items' enumeration numbers
Comma	,		Delimits sense block translation variants
			Comes after headword showing that it has additional information(ending or synonym)
Colon	:	Bold	Indicates that headword is not used by itself, but constitutes compound phrase that follows immediately
Round brackets	(and)		Embrace additional explanations or notes
		Only right one	Follows usage translations enumeration numbers
Dash	-	Followed by end of line symbol or tag	Line break
			Compound or united word
		Bold, preceded by space character or comma	Ending prefix in the form block

There are also reserved character constructions or/and abbreviations that also indicate or delimit some lexicographical units, see *Table 3.*

Table 3. Reserved characters and words

Character or word	Notation	Additional indicator	Construction	Function or meaning
Roman digits	I, II, II etc.	bold		Hyponym's number
Arabic digits	1, 2, 3 etc.	Bold, followed by full stop	1. | 2. | 3. etc.	Beginning of a sense item and its number
Arabic digits	1, 2, 3 etc.	Bold, followed by end round bracket	| 2) | 3) etc	Beginning of the usage item translation and its number
Kyrgyz abbreviations	See [7]	Italic, followed by full stop	[abbr].	Provide additional information or cut down repetitive text usage, see meanings in [7]

Relying on the above mentioned typographic, syntactic and reserved content features and their corresponding meanings it is possible to define some rules to delimit certain lexicographical units, see *Table 4.* This is the main principle of the parsing and annotation process which is described in the following section.

3. Method

When parsing formatted text data, there are three main features document needs to have to be parsed:

1. Layout
2. Formatting
3. Content patterns (repetitive text sequences)

Parsing is arranged relying on the one of this feature or combination of them since each of the mentioned features refers to a certain semantic unit of human readable textual data. For example, having bold word(s) in the beginning and out dented (shifted to the left) line most probably refer s to the next dictionary entry (here bold is a formatting feature and out dentation is a layout feature (see *Figure 1)).* Combinations of such kind of features give us opportunity to define rules to delimit certain lexicographic unit.

Table 4. Some rules for lexicographical units' delimitation

Delimiter	Notation	Additional feature	Delimits What
Comma + space + dash	, -		Headword and the suffix that changes headword ending
Colon	:	bold	First word from remainder part of a headword in complex headword cases
Comma	,		Semantically close meanings of a sense
Full stop	.		One sense from another sense; one translation from another translation

THE OBJECTIVE: to detect beginning and ending points of every lexicographical unit and mark it up with an appropriate XML tag.

Depending on pursued goals dictionary data may be encoded in different views. It depends on what kind of information is going to be captured via encoding. There are three main views (or information aspects) among others dealing with complexity of both typography and information structure.

- (a) the typographic view — the two-dimensional printed page, including information about line and page breaks and other features of layout
- (b) the editorial view — the one-dimensional sequence of tokens which can be seen as the input to the typesetting process; the wording and punctuation of the text and the sequencing of items are visible in this view, but specifics of the typographic realization are not
- (c) the lexical view—this view includes the underlying information represented in a dictionary, without concern for its exact textual form [3].

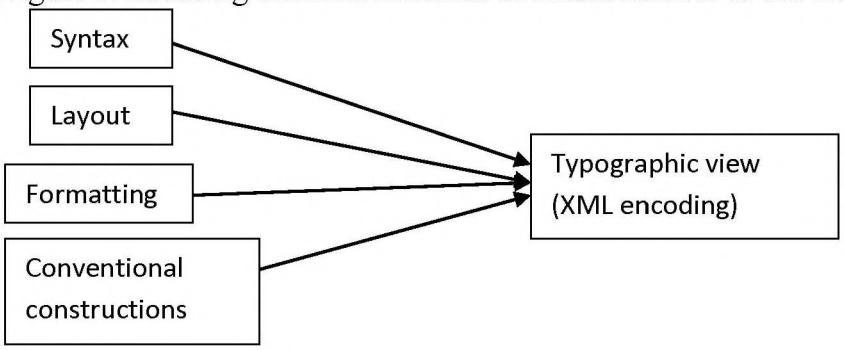
In this context our objective is to obtain editorial view of the dictionary. But before this, typographical view may simplify the whole task of annotation processes.

Pre-annotation Phase

Parser is intended to accept pre-annotated typographical view of the dictionary. Pre-annotation implies converting visually detectable typographical features to XML format. At the beginning it is easier to obtain typographic (not editorial or lexical) view in XML representation from formatted source text, because this procedure doesn't require any delimitation or parsing techniques, only converting layout, formatting and special purpose content (content indicators) into form of xml tags. For example, "some string in bold font" would be represented as "<bold> some string in bold font</bold>"; or to encode the order of lines every line might have its number: <line number = 10>the content of tenth line</line>; and content information such as abbreviations should be marked up by <abr/> tag etc. It is like an html code of a document, since html was originally

intended to reflect the document structure (but not semantics). We can use names of tags and attributes to encode any information we need. All these pre-annotated XML elements will be useful in main annotation, i.e. parsing and marking up processes. Pre-annotation allows reduce the number of different data types into one data type: linear XML (see *Fig.2.*). This process should preserve typographic information but in different representation.

Figure 5. Reducing different information media number to one linear XML.[8]



As a result of this simplification the code and logic of the parser becomes clearer, i.e. without excessive complexities of formatted text editors' inner data representation mechanisms. Having performed this reduction the programmer will need only XML processor (XSLT or other XML supporting languages) and a means to perform parsing of bare (without formatting) textual content. Data in the XML representation can be edited even in a simple text editor.

Parsing Technique Based on Delimitation & Markup

Entries Delimitation

The main issue of parsing was that not every dictionary entry captured exactly one paragraph. So another delimiter or delimitation mechanism had to be defined. The task of delimitation has various solutions. For example the most obvious solution would be to find every occurrence of line that is shifted to the left. This solution exploits layout feature of the typographical view, see *Table 1 and Figure 2.*

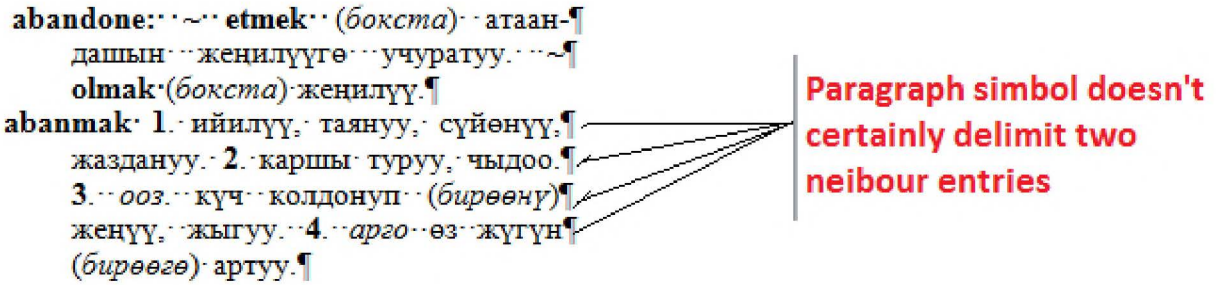


Figure 6. The main issue in entries delimitation task.

Another solution is to implement content-based parsing scenario where new delimiter (string of characters), i.e. text that certainly refers to the two articles meeting spot should be defined. Whatever parser's principle is the workflow of entries delimitation and basic structure delineation would be as following (see *Listing 2*): (In fact we chose the latter one, which defines new delimiter string)

Listing 1. Dictionary entries delimitation workflow pseudo code

```
1    Begin
2    For each line in Dictionary Do
```

[8] Conversion of punctuation characters to XML tags is rather optional

```
3     Select line
4     If line is out dented Then
5     (first line of the entry found)
6     Set cursor to the beginning of the line
7     put end tag of previous entry (e.g. </article>)
8     put start tag of next entry (e.g. <article>)
9     Else
10    Proceed to the next iteration
11    End IF
12    End For
13    Set cursor to the very beginning of Document
14    Cut the first occurrence of close entry tag(</article>)
15    Set cursor to the very end of Document
16    Paste close tag cut in 14
17    End
```

Note: In this algorithm the very first entry will be preceded by excessive end tag (</article>) and the very last entry will lack ending tag (</article>), that is why lines 13-16 are added.

Entry Parsing

Form Block Parsing

Now that until this moment delimitation is performed, i.e. every dictionary entry is delimited and marked up with <article > tags we can access each of them one by one. This may be realized either by means of XPath or by means of procedural programming language with or without XML support. Choice of solution depends on programmer's preferences. But whatever the tool is, headword of an entry should be recognized as the first word in the form block, and form block is the sequence of Turkish characters in bold font starting from the very beginning of the entries content. By traversing every entry and finding the word satisfying this condition it is possible to implement form block markup function, see *Listing 2*.

Listing 2. Pseudo code of the form block parsing and marking

```
1     Begin
2     For each entry in Dictionary Do
3     Select content
4     Find text which is bold & Turkish
5     If word is found & it is at the beginning of content
6     Then
7     Mark it up as form block(Form block's found)
8     Select first word
9     Mark it up as headword
10    Select remainder part of content
11    If remainder part is not empty
12    Mark remainder part as headword tail
13    Parse headword tail (see Listing 3.)
14    End If
15    Else
16    Throw an exception (entry syntax is wrong)
17    End If
18    End For
19    End
```

Not every headword consists of one word or it may have specific cases when it changes or it may have very close synonyms. So that the headword variations (possible syntaxes) are:

1. headword
2. headword, -ending
3. headword: ~ headword tail
4. headword, Synonym
5. HEADWORD (Acronym's Expanded Form'текст кыск.)

Description: 1. Simple headword; 2. Headword with ending; 3. Complex headword: one meaning but several words; 4. two or more words very close by meaning and by spelling; 5. Abbreviated word with the meaning of each capital letter and text in Kyrgyz containing reserved contraction 'кыск.' (contraction of 'abbreviation' like 'abbr.').

Actually this information may be found in form block and it should be parsed (parser is described in *Listing 3.*)

Listing 3. Headword tail parsing and markup (location: dictionary/entry/form/)

```
1   BEGIN
2   CAPTURE <tail>
3   GET content of <tail> as Content
4   IF Content is empty THEN
5   EXIT
6   ELSE_IF Content starts with ', -' THEN
7   MARKUP text after ', -'as </ending>
8   UNMARK <tail/>
9   ELSE_IF Content starts with ':' THEN
10  MARKUP text after ': 'as <headword part=2/>
11  CAPTURE <headword>
12  RENAME <headword> to <headword part=1>
13  CAPTURE <tail/>
14  UNMARK <tail/>
15  ELSE_IF Content starts with ',' THEN
16  MARKUP text behind ', 'as <synonym/>
17  UNMARK <tail/>
18  ELSE_IF Content starts with ' (' THEN
19  MARKUP text behind ' ('as <acronym/>
20  UNMARK <tail/>
21  ELSE
22  Throw an exception (entry syntax is wrong)
23  END_IF
24  END
```

Definition Block Parsing (see Results)
Sense Block Parsing (see Results)
Usage Block Parsing (see Results)

4. Discussion

Input File Problem

Initially dictionary was available in pdf format. But since it was difficult to process text data in pdf file directly (or maybe we lack knowledge on how to work with it) it was decided to convert it to more text accessible formats like txt, html or doc. That is why AdobeAcrobat's (version 11.0.2) pdf converter was used, which in fact offers various types of files to save as, e.g. eps, html, docx, doc, pptx, xlsx, xml table, txt etc. Of course the most interesting format was xml, since we were

going to obtain such an encoding of data that would provide us with explicitly structured semantic units of the dictionary. As turned out AdobeAcrobat's xml file didn't preserve all the information we needed for parsing. That is why the most close to origin file doc format was chosen. Here beneath in the *Table 1*, we showed criteria which we took as a guide to choose a file type to start up with:

Table 5. AdobeAcrobat's output files comparison with regard to dictionary data completeness

File type	Formatting	Layout	Content
Doc	Preserved	Preserved	Preserved
Xml	Lost	Lost	Preserved

The only shortcoming of doc file was that it contained some misspellings or precisely saying character mis-encodings when Turkish character occurred in Kyrgyz word or vice versa. Probably this happened as a result of OCR reading when it attempted to read word with one encoding but the word was actually in different, e.g. in Turkish(Latin character set) word "aba" first character is in Cyrillic. This issue was fixed programmatically as part of the input file normalization. Another task was to remove all data except dictionary entries: introduction, usage notes, headings, page numbers etc.

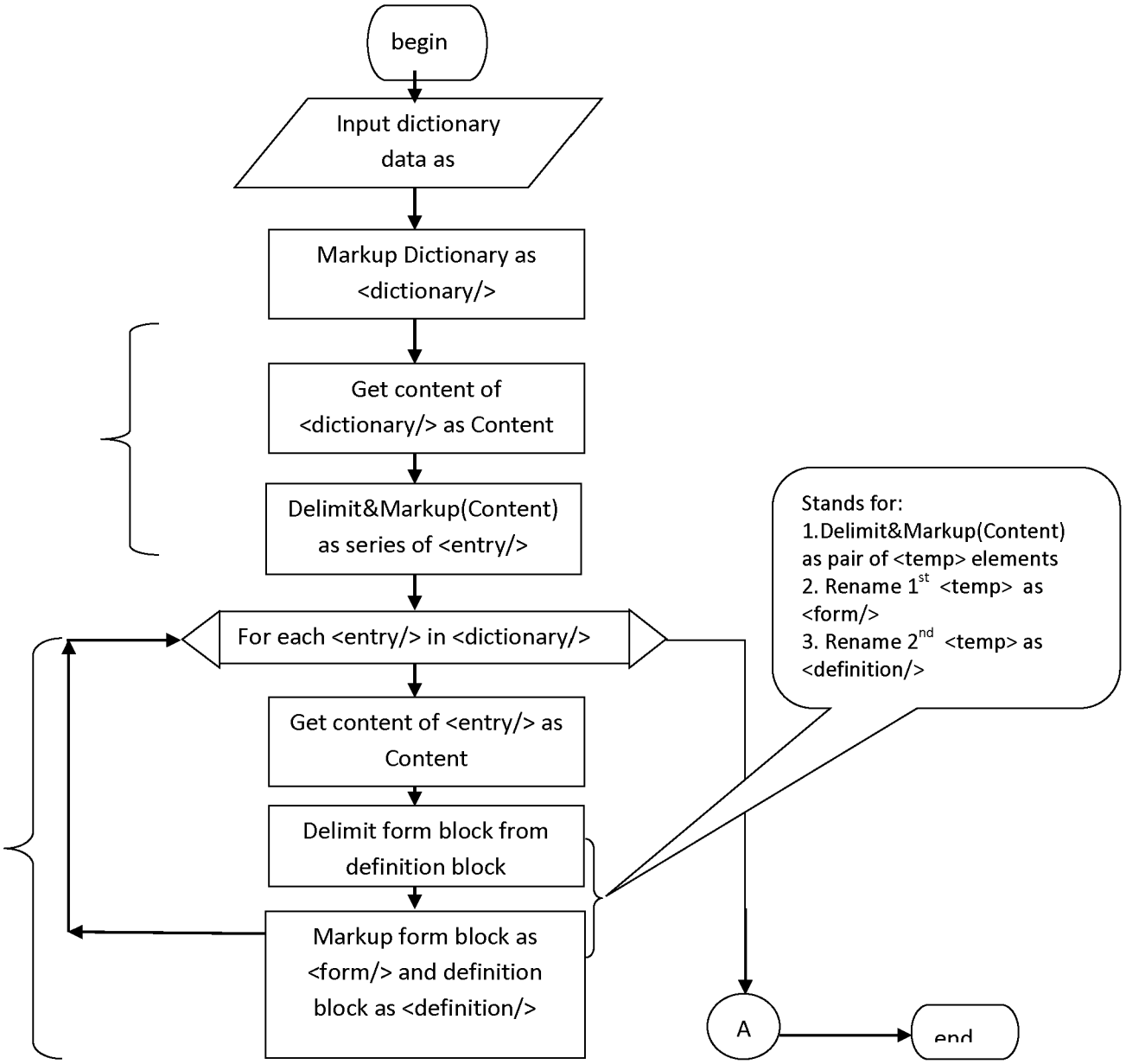
Ambiguity Problem

Any sense item including the last one theoretically may have a sense example. If so and if example and usage item are delimited in similar way there is no way to identify the item coming after the last sense weather it is example or usage item.

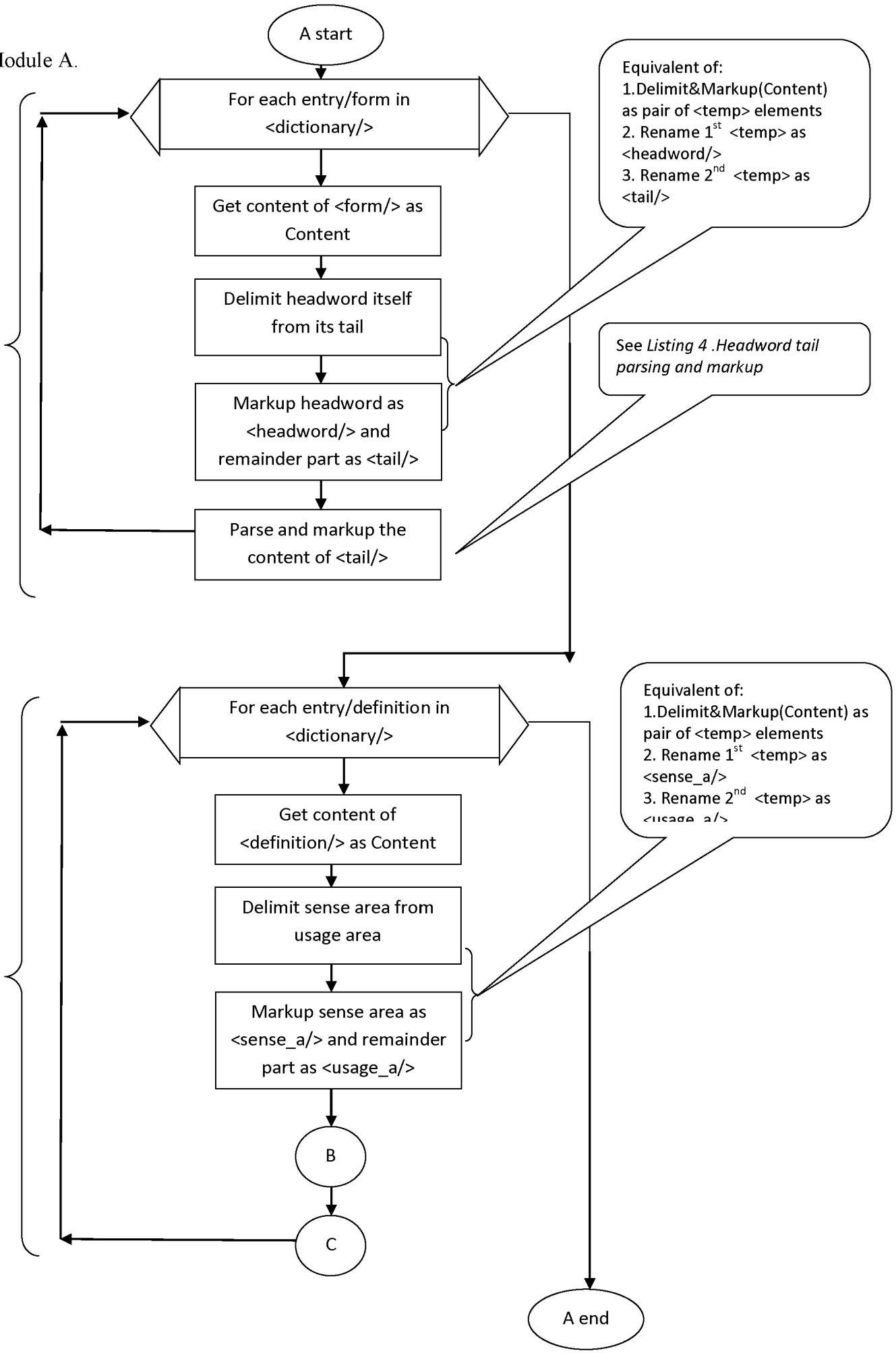
5. Results:

An Algorithm of the Parser. The main result of the work is the workflow of the Cumakunova's dictionary parser, see *Figure 5.* Node A is expanded in Module A, and nodes B and C in corresponding Modules B and C

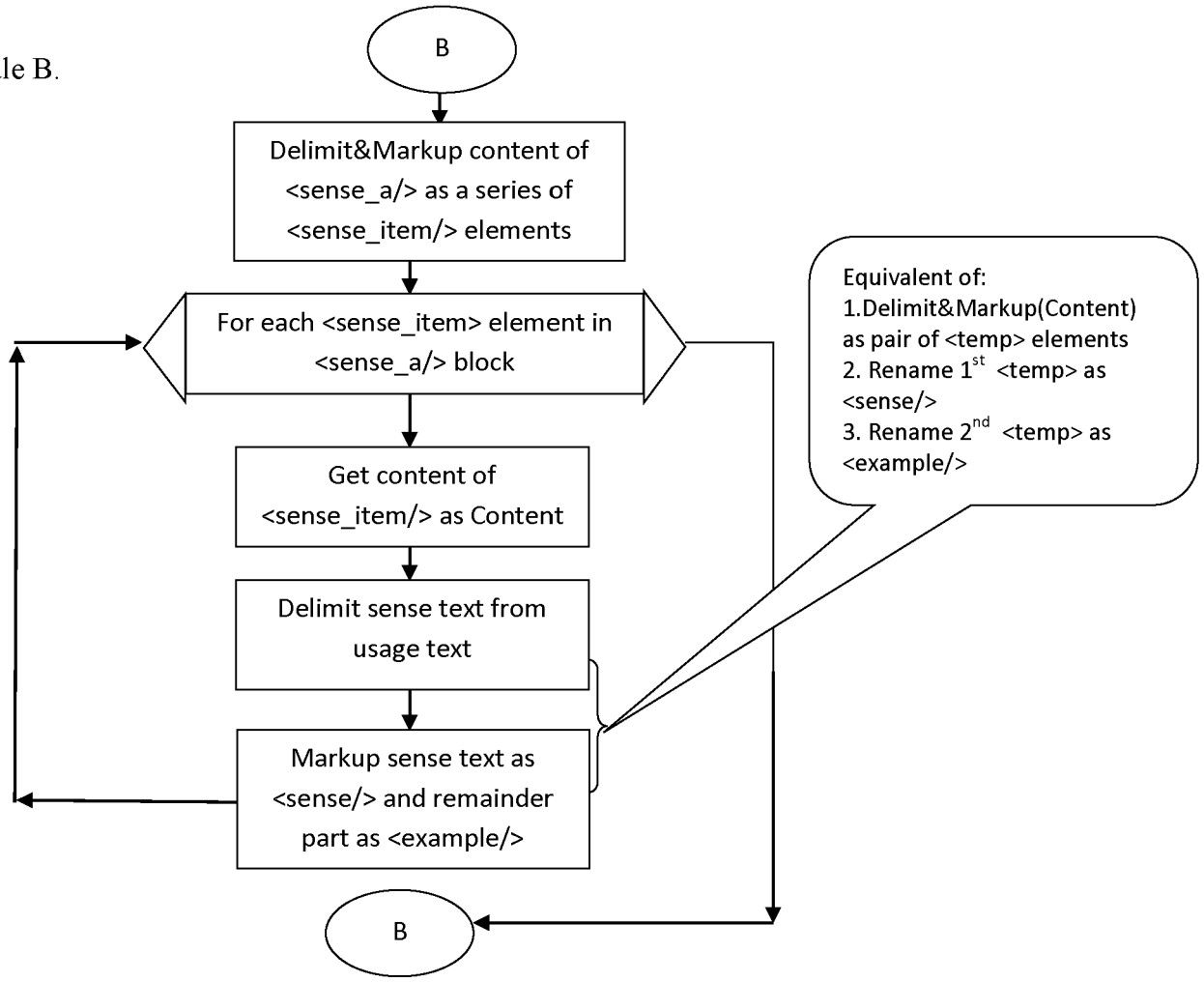
Figure 7. The workflow of the parser


begin
Input dictionary data as
Markup Dictionary as <dictionary/>
Get content of <dictionary/> as Content
Delimit&Markup(Content) as series of <entry/>
For each <entry/> in <dictionary/>
Get content of <entry/> as Content
Delimit form block from definition block
Markup form block as <form/> and definition block as <definition/>
A
end
Stands for:
1.Delimit&Markup(Content) as pair of <temp> elements
2. Rename 1st <temp> as <form/>
3. Rename 2nd <temp> as <definition/>

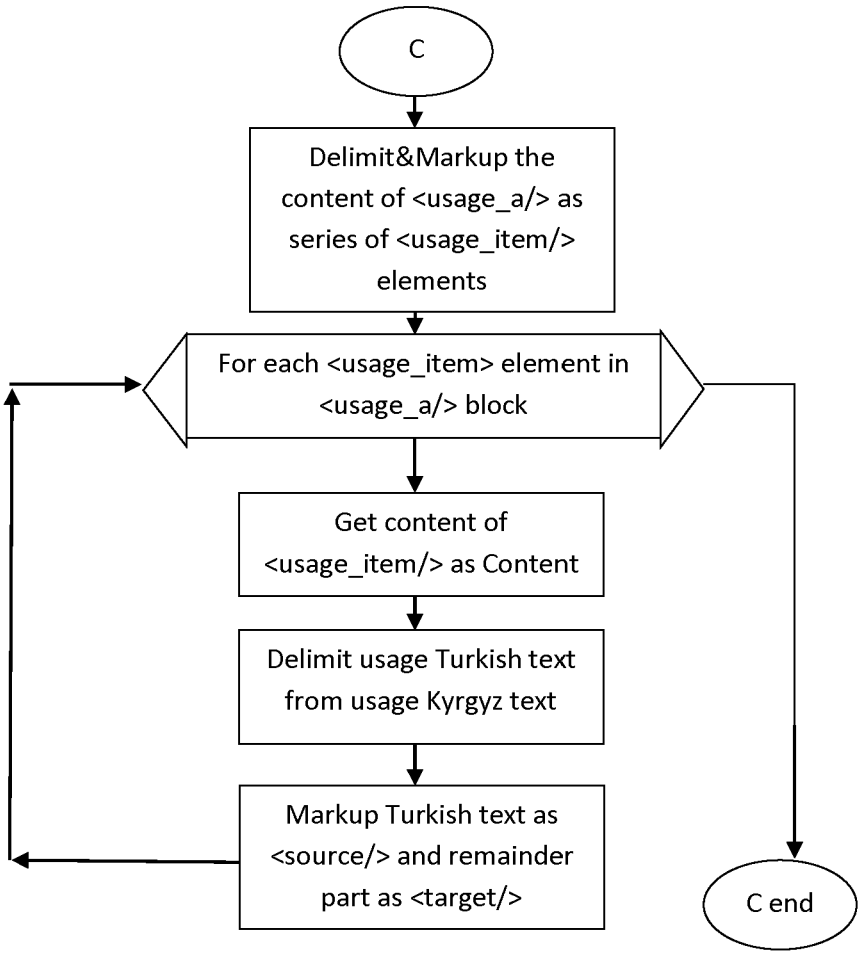
Module A.


A start
For each entry/form in <dictionary/>
Get content of <form/> as Content
Delimit headword itself from its tail
Markup headword as <headword/> and remainder part as <tail/>
Parse and markup the content of <tail/>
Equivalent of:
1.Delimit&Markup(Content) as pair of <temp> elements
2. Rename 1st <temp> as <headword/>
3. Rename 2nd <temp> as <tail/>
See Listing 4 .Headword tail parsing and markup
For each entry/definition in <dictionary/>
Get content of <definition/> as Content
Delimit sense area from usage area
Markup sense area as <sense_a/> and remainder part as <usage_a/>
Equivalent of:
1.Delimit&Markup(Content) as pair of <temp> elements
2. Rename 1st <temp> as <sense_a/>
3. Rename 2nd <temp> as
B
C
A end

Module B.



Module C



Applications

Parsed data has been applied to obtain StarDict's Textual file dictionary bases and to create PhpMyAdmin (MySQL administration tool written in PHP) xml import file. But it should be stated that this is only an intermediate stage whence primary goal of the parental research is TEI standard which allows for creating reliable ontologies.

StarDict Dictionary. Stardict has its Textual file dictionary format that uses RELAX NG schema. According to StarDict creator, Hu Zheng, Textual file format was designed to reflect structure of a dictionary [4]. This textual representation of a dictionary has several advantages against earlier created binary opaque format that hides source data and doesn't allow editing dictionaries.

Textual file format may be used to:

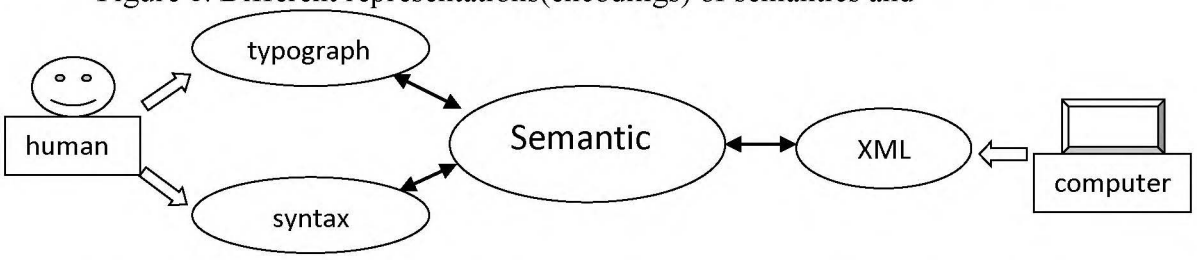
1. examine dictionary content
2. make changes to a dictionary
3. create a new dictionary from scratch

It should be said that binary files such as images, audio, video are not stored in Textual file directly, instead their links refer to the resource location. The order of articles (or dictionary entries) in this format doesn't matter.

6. Conclusion

This study of data structuring has shown that semantic structure of a dictionary as of any other abstraction is nothing but different encodings weather by means of typography(layout, formatting, text) and syntax or by means of computer purposed data representation language such as XML (see *Figure 4*). In the first case human beings are able to decode the essential meaning of the data since they are aware of typography-to-semantics interpretation rules. For example in case of Cumakunova's dictionary, when they see bold text they get the knowledge that this text is in Turkish and when they see normal text it must be in Kyrgyz; or when they see out dented line they understand that this is the beginning of a new dictionary entry etc. To enable machines/computers to operate with the same semantics the human encodings must be converted to machine-readable format, i.e. information about any semantic unit must be provided explicitly. Computer must be provided by names, attributes and boundaries of units in order to know their semantics and structure. XML technology provides this meta-information enabling computers to access, process and operate this kind of structured information at the semantic level not on physical.

Figure 8. Different representations(encodings) of semantics and



Efficient global information exchange is impossible without commonly understood and shared standards and concepts. Big amounts of information have to be presented as much as possible in unambiguous and precise manner. This goal was pursued in efforts of parsing and annotation of Turkish-Kyrgyz dictionary, since large text corpora from different epochs have to be parsed and annotated for performing further analysis, such as building a meta– lemma list for the project interdependencies between language and genomes [3].

References

1. Dieter Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag Berlin Heidelberg GmbH, 2004.

2. Dieter Fensel: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce (Draft version). Springer-Verlag Berlin Heidelberg GmbH, 2004. Page 50. Retrieved from: http://software.ucv.ro/~cbadica/didactic/sbc/documente/silverbullet.pdf, accessed 7/28/2016

3. TEI CONSORTIUM, eds.: TEI P5: Guidelines for Electronic Text Encoding and Interchange. http://www.tei-c.org/Guidelines/P5/

4. Hu Zheng,: Textual Dictionary File Format. StarDict project on GitHub, https://github.com/huzheng001/stardict-3/blob/master/dict/doc/TextualDictionaryFileFormat, accessed 8/6/2016

5. Гүлзура Жумакунова: Түркчө-Кыргызча сөздүк, 50 000 сөз. Кыргыз-Түрк «Манас» Университетинин басылмалары, Бишкек, 2005.

6. Гүлзура Жумакунова: Түркчө-Кыргызча сөздүк, 50 000 сөз. Кыргыз-Түрк «Манас» Университетинин басылмалары, Бишкек, 2005. pages 28-29

УДК 510.5, 519.768.2

МЕТОДОЛОГИЯ АВТОМАТИЗИРОВАННОГО ПОПОЛНЕНИЯ СЛОВАРЯ СИСТЕМЫ МАШИННОГО ПЕРЕВОДА ДЛЯ КАЗАХСКО-РУССКОЙ И КАЗАХСКО-АНГЛИЙСКОЙ ЯЗЫКОВОЙ ПАРЫ

У.А. Тукеев, *Казахский Национальный Университет имени аль Фараби, Алматы, Казахстан. ualsher.tukeyev@gmail.com*
Д.Р. Рахимова, *Казахский Национальный Университет имени аль Фараби, Алматы, Казахстан.di.diva@mail.ru*
Ж.М. Жуманов, *Казахский Национальный Университет имени аль Фараби, Алматы, Казахстан. z.zhake@gmail.com*

Аннотация: В статье описывается методология автоматизированного пополнения словаря системы машинного перевода Apertium для казахско-русской и казахско-английской языковой пары. Цель данной методологии состоит в оказании помощи пользователю обнаружить лучшую морфологическую парадигму в одноязычном морфологическом словаре Apertium. Приведены практические результаты.

Ключевые слова: заполнение словарей, Apertium, казахский, русский и английский язык.

METHODOLOGY OF THE AUTOMATED ENRICHMENT OF MACHINE TRANSLATION SYSTEM DICTIONARIES FOR KAZAKH-RUSSIAN AND KAZAKH-ENGLISH LANGUAGE PAIR

U.A. Tukeyev, *Al Farabi Kazakh National University, Almaty, Kazakhstan. ualsher.tukeyev@gmail.com*
D.R.Rakhimova, *Al Farabi Kazakh National University, Almaty, Kazakhstan. di.diva@mail.ru*
Zh.M. Zhumanov *Al Farabi Kazakh National University, Almaty, Kazakhstan. z.zhake@gmail.com*

Abstract: This paper describes the methodology of the automated enrichment dictionary of the machine translation system Apertium for the Kazakh-Russian and Kazakh-English language pair. The purpose of this methodology consists in assistance to the user to find the best morphological paradigm in the monolingual morphological Apertium dictionary. Practical results are presented.

Keywords: filling of dictionaries, Apertium, Kazakh, Russian and English.

Введение

На данный момент существует много различных словарей, как печатные, так и электронные. Словари, используемые в машинном переводе (МП) может содержать переводы на различные языки сотен тысяч слов и фраз, а также предоставить пользователям