Türk Dilleri Araştırmaları, 21.2 (2011): 235-249

Suffix Identification in Turkic Texts

Münevver Tekcan[*]
(Kocaeli)

Özet: Türkçe metinlerin linguistik özelliklerini incelemek için, bir gramatikal dizin hemen hemen şarttır. Basit bir gramatikal dizin, transkripsiyonlu olarak her kelimeyi, bulunduğu yeriyle birlikte metinde listeler. Daha detaylı analiz için, sıralanmış son eklere ve kelimenin kullanıldığı içeriğe ihtiyaç duyulur. Daha önce, bir el yazmasının çeviriyazısından basit bir dizin oluşturmak için başarılı bir şekilde çalışan bir yazılım geliştirmiştim. Bu başarı beni, daha ileri gitmem ve son ekleri tanımlayan, çözümleyen (ayıran) gramatikal bir dizin oluşturmam yönünde cesaretlendirdi. Bu makale, örneklerle bu süreci açıklamaktadır. Bu yazılım, gramatikal dizinle el yazması arasında doğrudan doğruya yayımda kullanılabilecek nitelikte, kesintisiz bir ara yüz sağlamaya çalışmaktadır.

Anahtar Sözcükler: Gramatikal dizin, yazılım, son eklerin tanımlanması, Türkçe metinler.

Abstract: In order to examine the linguistic features of Turkic texts, a grammatical index is almost essential. A simple grammatical index, lists each word in the text with its position in the transcription. For analysis that is more detailed ordered suffixes and the context in which the word is used is needed. I had previously developed a software to produce a simple index from a transcription of a manuscript. This was successful and provided encouragement to go further and try to produce a grammatical index with suffix identification and breakdown. The paper, explains this process with examples. The software, tries to provide a seamless interface between manuscript and grammatical index of a quality that can be used directly in a publication.

Keywords: Grammatical Index, software, suffix identification, Turkic texts

[*] Kocaeli Üniversitesi, mtekcan@kocaeli.edu.tr

Introduction

Grammatical indexes are an important tool in the study of Turkic texts. These indexes identify suffixes and where they are used in a text. While this paper concentrates on the identification of Inflection Suffixes, the method used should work on Derivational Suffixes as well.

Grammatical indexes can be produced by hand, but in the age of Information Technology and the Internet, automatic generation of grammatical indexes is now possible. For a number of years, I have been working on the development of a project to generate grammatical indexes automatically.

Overview

Figure 1

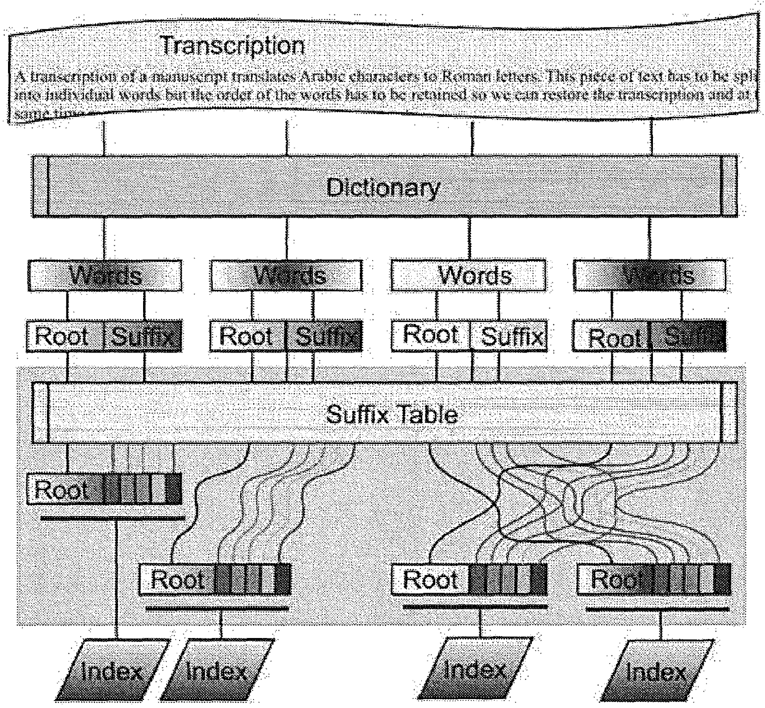

Transcription
A transcription of a manuscript translates Arabic characters to Roman letters. This piece of text has to be spl
into individual words but the order of the words has to be retained so we can restore the transcription and at
same time
Dictionary
Words
Words
Words
Words
Root
Suffix
Root
Suffix
Root
Suffix
Root
Suffix
Suffix Table
Root
Root
Root
Root
Index
Index
Index
Index

Figure 1 shows an overview of a website that I am developing. At the top of this diagram is transcribed Turkic text written in non-Latin alphabets. The original text could be prose or poetry, or a mixture of the two. Besides the text itself, the transcription includes index information such as canto and verse number, folio and line number, which is relevant to the manuscript or other types of numbering such as sentence and paragraph number.

The program reads transcribed text in Microsoft Word format. The transcribed text is split into individual words and then into root and suffix string. A special dictionary is used to split individual words into root and suffix string. The suffix string is further divided into individual suffix tokens using a suffix table, which is the topic of this paper. All the way through these processes, links to the transcription are retained, and therefore it is possible to produce a number of types of index. For example:

Root and suffix, referenced with line and page number
Root and suffix, referenced with line and folio number
Suffix, referenced with line and page number
Suffix, referenced with canto and verse number

Dictionary

A dictionary is implemented using a MySQL data base. This provides a sophisticated platform to exploit the facilities of a database such as to select and to perform complex searches.

Each entry in the dictionary has information on:

The root word itself
The word's origin
Definition
A flag indicating noun, proper name or verb

With regard to suffix identification, this dictionary can identify root words and mark them as noun or verb. This is important in selecting which suffix list to use.

Table 1

Word in transcription	Root word in dictionary	Root and suffix string
bolmadıŋ	bol-	bol-madıŋ
ḳıladur	ḳıl-	ḳıl-adur
cemālıŋa	cemāl	cemāl+ıŋa

Table 1 shows a short list of root and suffix examples.

Suffix Breakdown

There are several reasons for breaking down the suffix strings into a list of separate suffix tokens:

Individual suffix strings provide semantic information and hence give meaning to the text under investigation.

Suffix identification helps with the grammatical classification of words.

Under Arat's[1] ordering, individual suffix tokens have a particular weight or rank. This rank defines the order in which suffixes should appear in any index. Therefore, suffix identification is crucial to providing the correct ordering of suffixes.

Suffix lists and the sequences of Turkic suffixes are central to the software used in the website. Suffix lists and their ordering are converted to BNF[2] as a starting point to write the software and to ensure that there is a close correspondence between the suffix structures used in the Turkic languages and the structure of the program. Suffix ordering operates by creating tree structures where the nodes in the tree represent suffix tokens and the branches represent ordering. Trees are created from the top down only when a path to the root has been found.

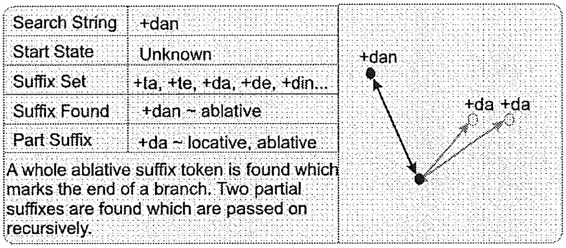
During the identification of individual suffix tokens, the software moves from one state to another where each state is defined by a set of valid suffix tokens. At each state in the suffix identification process, each suffix in the set of valid suffixes is compared with the suffix string that is being broken down. When a suffix token or key is found at the beginning

[1] Cf. Arat 1979: V-XII.

[2] See Naur 1960.

of the string to be broken down, a state change can take place. If not, a null state is entered which in effect terminates progress through the tree and removes all paths leading to failure.

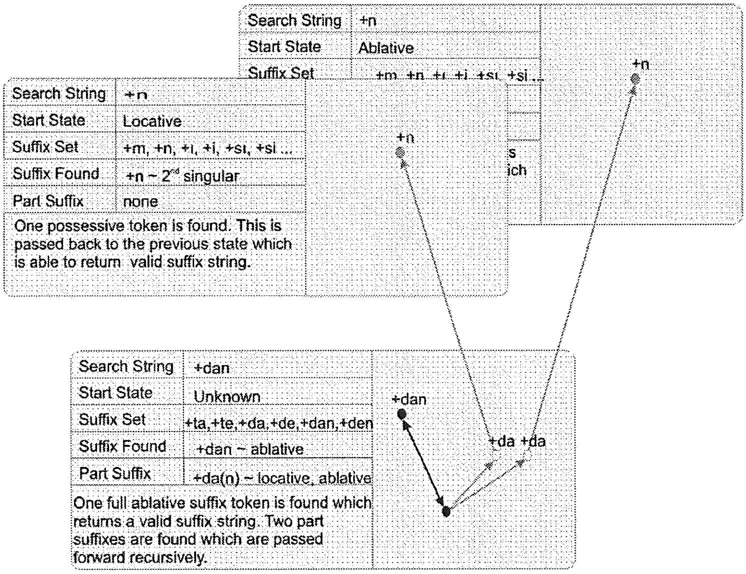
For example, in trying to break down the suffix string +*dan*, initially a comparison is made with all the sets of valid suffix strings that can start a suffix. This search identifies the ablative suffix +*dan* and two other suffixes with the leading suffix +*da* and leaving +*n* and +*n*.



Search String	+dan
Start State	Unknown
Suffix Set	+ta, +te, +da, +de, +din...
Suffix Found	+dan ~ ablative
Part Suffix	+da ~ locative, ablative

A whole ablative suffix token is found which marks the end of a branch. Two partial suffixes are found which are passed on recursively.

Figure 2



Search String	+n
Start State	Ablative
Suffix Set	+m, +n, +ı, +i, +sı, +si ...

Search String	+n
Start State	Locative
Suffix Set	+m, +n, +ı, +i, +sı, +si ...
Suffix Found	+n ~ 2nd singular
Part Suffix	none

One possessive token is found. This is passed back to the previous state which is able to return valid suffix string.

Search String	+dan
Start State	Unknown
Suffix Set	+ta,+te,+da,+de,+dan,+den
Suffix Found	+dan ~ ablative
Part Suffix	+da(n) ~ locative, ablative

One full ablative suffix token is found which returns a valid suffix string. Two part suffixes are found which are passed forward recursively.

Figure 3

The next stage shows the process of following the two partial nodes found previously. Two searches identify *+n* as the 2nd person singular possessive suffix. Overall the program has found three matches, an ablative suffix, an ablative 2nd person singular and a locative 2nd person singular suffix. The program can now return back through itself, creating a tree structure from the top down.

Start state	Ablative State	Possessive State
root**miz** ⇨		
root**dedi-ya** ⇨	root**di-ya** ⇨	
root**tindin** ⇨		
root**dan** ⇨	root**n** ⇨	root⇨
root**tan**⇨		
	Dative State	
root**tege** ⇨	root**ge** ⇨	
root**ta** ⇨	root**ta** ⇨	root ⇨
root**ken** ⇨	root**n** ⇨	root ⇨
root**dan** ⇨		
root**ran**⇨	root**n** ⇨	root ⇨

Figure 4

This diagram tries to show how a finite state machine can work. Basically it is seen as a series of filters. The ablative filter will only let through suffixes in the set that identifies the ablative state. In this case {*+tın*, *+tin*, *+dın*, *+din*, *+ta*, *+te*, *+da*, *+de*, *+dan*, *+den*}. The possessive state is similarly defined by a set of suffix tokens {*+(x)m*, *+(x)ŋ*, *+(s)ı*, *+(s)i*, *+(x)mız*, *+(x)miz*, *+(x)ŋız*, *+(x)ŋiz*, *+(x)ları*, *+(x)leri*}. In this example tokens can only pass through the dative or ablative states before they can become possessive.

Suffix Lists

Lists of suffix tokens for verbs and nouns, based on Arat's grammatical ordering, are used in suffix identification. There are separate entries for each suffix type and conjugation and entries for cases such as buffers between vowels and character changes such as, *d* > *t* in the past tense clauses. Suffix tokens are stored as an ordered list of objects.

Suffix identification and extraction should not be seen in isolation but in a wider context as shown in Figure 1. Whatever happens in the identification and extraction process, each word is linked to the transcription enabling the generation of an index.

Each instantiation of a suffix token contains:

An index for the objects position in the list

The suffix token

The suffix type or value in a finite state diagram

Auxiliary variable (currently used for conjugation)

Suffix rank

Example instantiation of a suffix token

```
$nounsuffix[$p++] = new Suffix("m" , possessive, 1, $p);
$nounsuffix[$p++] = new Suffix("im", possessive, 1, $p);
$nounsuffix[$p++] = new Suffix("im", possessive, 1, $p);
$nounsuffix[$p++] = new Suffix("um", possessive, 1, $p);
$nounsuffix[$p++] = new Suffix("üm", possessive, 1, $p);
```

Table 2

As the suffix tokens are searched for, the search may go through a number of state changes, such as noun to plural to possessive. The recursive function uses a set of heuristics to ensure that the search only goes through valid state changes. For example, for a noun, a plural case can only follow a noun; it cannot follow any other cases.

Ordering Suffix Tokens

Suffix token ordering is based on Arat's grammatical ordering

The suffix table is central to ordering suffix strings

Each suffix token is given a rank or weight

Ranks are concatenated if there is more than one token in the suffix string

The suffix tables contain an ordered list of all possible tokens. The ordering of this list is the same as the order in which they would appear in a grammatical index. Heuristics are hard coded to handle special cases that cannot be defined in suffix table structure.

Single Token:

Past	**ye**-di	*he ate*	rank = 048155*
Ifinitive	**ye**-mek	*to eat*	rank = 171
	order in grammatical index		rank derived from suffix table

Table 3 **Comparisons are alphabetical*

Multiple Tokens:

PAST	1st Person Singular	**ye**-di-m	*I ate*	rank = 048151
	2nd Person Singular	**ye**-di-n	*you ate*	rank = 048154
	3rd Person Singular	**ye**-di*	*he ate*	rank = 048155
		order in grammatical index		from suffix table

Table 4 * The 3rd person singular of the past tense has an empty suffix

My Index Page (MIP) comparisons

	Concordance	**Work Bench**	**VATEC**	**MIP**
Read Microsoft Word Files	✗	✗	✗	✓
Transliteration	✗	✗	✓	✗
Exact Transcription	✗	✗	✓	✗
Transcription	✗	✗	✓	✓
Morpheme	✗	✗	✓	✓
Definition	✗	✗	✓	✓
Origin	✗	✗	✗	✓
Case	✗	✗	✓✓	✓✓
Translation	✗	✗	German some English	✗
Sentence parsing	✗	✓	A little	✗
Alphabetic index	✓	✗	✗	✓
Context text	✓✓	✗	✓	✓
Write Microsoft Word files	✗	✗	✗	✓
Statistics	✓	✓✓	?	✓

Table 5

Table 5 compares my index pages with three other possible solutions to a grammatical index. The final figure at the end of this paper shows my index pages as part of a methodology to transcribe, process and understand Turkic texts. Feedbacks from domains external to the technology facilitate suffix identification. Other solutions provide limited information that tends to be less integrated with the problem.

In comparing my suite of programs with other programs, there does not seem to be any that are trying to do exactly what I wanted to do. That is, to provide automatic grammatical index generation that can be used in a publication, from a transcription that would be part of the same publication. The VATEC programs produce a detailed breakdown of each sentence with good identification of each case, but there is no index. Concordance programs produce good context information and in alphabetical order, but there is no grammatical information. Programs similar to the Writers Work Bench produce a lot of statistics and do a certain amount of sentence parsing to produce readability scores.

Conclusion

I believe that this method should work in practice for nearly all cases. I have successfully used the software to produce a number of works. Compared to the manual method, it is much quicker. Therefore, more time is available to concentrate on other linguistic aspects. Currently, the software is going through development so that it can run under an updated platform. The new software should be able to provide statistical analysis related to the output generated by suffix identification. Combining all facilities, the software should be able to:

Provide special fonts[3] for characters not in the standard character set
Read a transcription from a Microsoft Word document
Create a grammatical index in a Microsoft Word document
Provide a wide range of indexing options
Provide improved context information
Provide improved dictionary information
Provide improved suffix information
Sort suffixes according to Arat's ordering
Provide statistics
Produce a useable index from a working transcription with the minimum of editing

[3] Based on utf8 but providing dotted d, s, t and z characters.

Using information technology, the Internet and the software outlined in this paper has created a new methodology of studying Turkic texts. This involves many issues that will be discussed elsewhere; diagrammatically represented in the figure at the end of this paper, which shows the student integrated with the subject, the process of studying and the tools used to study.

Sample Grammatical Indexes

Example 1

Grammatical index providing root word, derivation, definition, suffix, suffix breakdown (highlighted for clarity) and line number.

amraḳ oġlum ne üçün busuşluġ keltiŋiz? tegin ḳaŋı ḳanḳa inçe tep
ötünti ıġlayu bu ne emgeklig yer ermiş neglük
toġdum men ḳaŋı ḳan inçe tep ayıttı neke ıġlayu buşuşluġ keltiŋ?
tegin inçe tep ötünti taştın ilinçüke önmiş ertim
üküş yoḳ çıġay emgeklig tınlıġlaraġ körüp ıġladım ḳanı ḳan inçe tep
yarlıḳadı[4]

[4] J. R. Hamilton, pp. 12-13.

amraḳ Sevilen, sevgili

a. 1

ayıt < ayt- < ay-(X)t- Söylemek

a. -tı dum [past(1st sin.)] 2

busuşluġ < busuş+lXgKederli, üzgün

b. 1

b. 2

çıġay Yoksul

ç. 3

emgeklig < emgek+lXg Zahmetli

e. 1, 3

ėr- İmek, olmak

e. -mişdum [past(3rd sin.)] 1

e. -tim dum [past(1st sin.)] 2

ıġla- Ağlamak

ı. -dım -dum [past(1st sin.)] 3

ı. -yu -yu [gerund] 2, 1

ilinçü Eğlence

i. +ke +ke [dative] 2

inçe Şöyle, şu şekilde; öyle

i. 3, 2, 1, 2

ḳan Han

ḳ. 3, 2

ḳ. +ı ı [possesive(1st sin.)] 3

ḳ. +ḳa +ḳa [dative] 1

ḳaŋ Baba

ḳ.+ı 1, 2

kel- Gelmek

k. -tiŋ -tiŋ [past(2nd sin.)] 2

k. -tiŋiz -tiŋiz [past(2nd plu.)] 1

kör- NULL Görmek

k. -üp -üp [gerund] 3

men NULL Ben

m. 2

ne NULL Ne

n. 1, 1

neglük NULL Niçin, hangi sebeple

n. 1

neke NULL Niye, niçin

n. 2

oġ(u)l Oğul

o. +um -um [possesive(1st sin.)] 1

ön- Çıkmak

ö. -miş -miş [past(1st sin.)] 2

ötün- Arz etmek, dilemek

ö. -ti ti [past(3rd sin.)] 2, 1

taş NULL Dış, dışarı

t. +tın tın [ablative] 2

te- Demek, söylemek

t. -p p [gerund] 3, 2, 2, 1

tegin < Çin Prens

t. 1, 2

tınlıġ < tın+lXg Canlı

t. +laraġ [plural accusative]3

toġ- Doġmak

t. -dum dum [past(1st sin.)] 2

üçün İçin, olduğundan

ü. 1

üküş < ük-(X)ş Pek çok

ü. 3

yarlıḳa- < yarlık+A- Buyurmak

y. -dı dı [past(3rd sin.)] 3 **yer** Yer

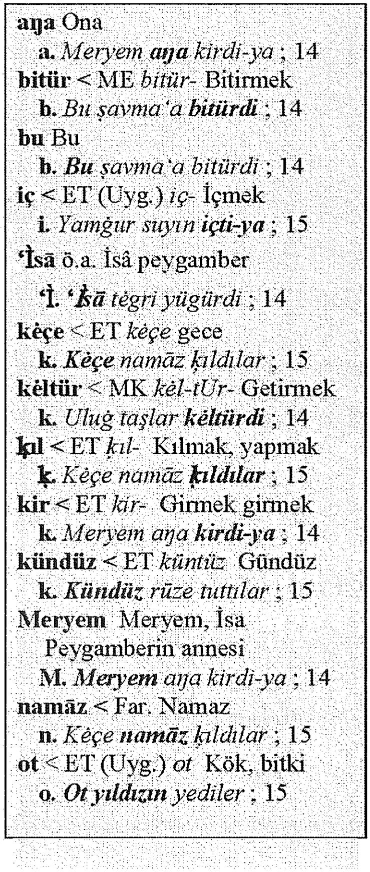
y. 1

Example 2

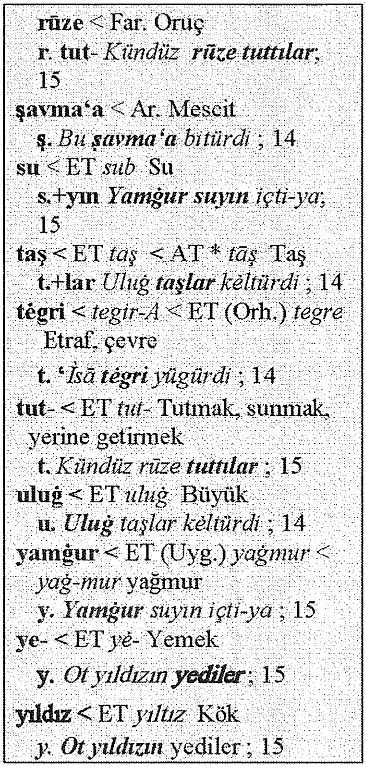
Grammatical index providing root, root word, derivation, suffix, context text with key word shown in bold and verse number.

14
ʻİsā tėgri yügürdi
Uluġ taşlar kėltürdi
Bu ṣavmaʻa bitürdi
Meryem aŋa kirdi-ya

15
Kündüz rūze tuttılar
Kėçe namāz ḳıldılar
Ot yıldızın yediler
Yamġur suyın içti-ya[1]



aŋa Ona
a. *Meryem* ***aŋa*** *kirdi-ya* ; 14
bitür < ME *bitür-* Bitirmek
b. *Bu ṣavmaʻa* ***bitürdi*** ; 14
bu Bu
b. ***Bu*** *ṣavmaʻa bitürdi* ; 14
iç < ET (Uyg.) *iç-* İçmek
i. *Yamġur suyın* ***içti-ya*** ; 15
ʻİsā ö.a. İsâ peygamber
ʻİ. ***ʻİsā*** *tėgri yügürdi* ; 14
kėçe < ET *kėçe* gece
k. ***Kėçe*** *namāz ḳıldılar* ; 15
kėltür < MK *kėl-tUr-* Getirmek
k. *Uluġ taşlar* ***kėltürdi*** ; 14
ḳıl < ET *ḳıl-* Kılmak, yapmak
ḳ. *Kėçe namāz* ***ḳıldılar*** ; 15
kir < ET *kir-* Girmek girmek
k. *Meryem aŋa* ***kirdi-ya*** ; 14
kündüz < ET *küntüz* Gündüz
k. ***Kündüz*** *rūze tuttılar* ; 15
Meryem Meryem, İsa Peygamberin annesi
M. ***Meryem*** *aŋa kirdi-ya* ; 14
namāz < Far. Namaz
n. *Kėçe* ***namāz*** *ḳıldılar* ; 15
ot < ET (Uyg.) *ot* Kök, bitki
o. ***Ot yıldızın*** *yediler* ; 15



rūze < Far. Oruç
r. tut- *Kündüz* ***rūze tuttılar***; 15
ṣavmaʻa < Ar. Mescit
ṣ. *Bu* ***ṣavmaʻa*** *bitürdi* ; 14
su < ET *sub* Su
s.+yın ***Yamġur suyın*** *içti-ya*; 15
taş < ET *taş* < AT * *tāş* Taş
t.+lar *Uluġ* ***taşlar*** *kėltürdi* ; 14
tėgri < *tegir-A* < ET (Orh.) *tegre* Etraf, çevre
t. *ʻİsā* ***tėgri*** *yügürdi* ; 14
tut- < ET *tut-* Tutmak, sunmak, yerine getirmek
t. *Kündüz rūze* ***tuttılar*** ; 15
uluġ < ET *uluġ* Büyük
u. ***Uluġ*** *taşlar kėltürdi* ; 14
yamġur < ET (Uyg.) *yaġmur* < *yaġ-mur* yağmur
y. ***Yamġur*** *suyın içti-ya* ; 15
ye- < ET *yė-* Yemek
y. *Ot yıldızın* ***yediler*** ; 15
yıldız < ET *yıltız* Kök
y. ***Ot yıldızın*** yediler ; 15

[1] TEKCAN, s. 190.

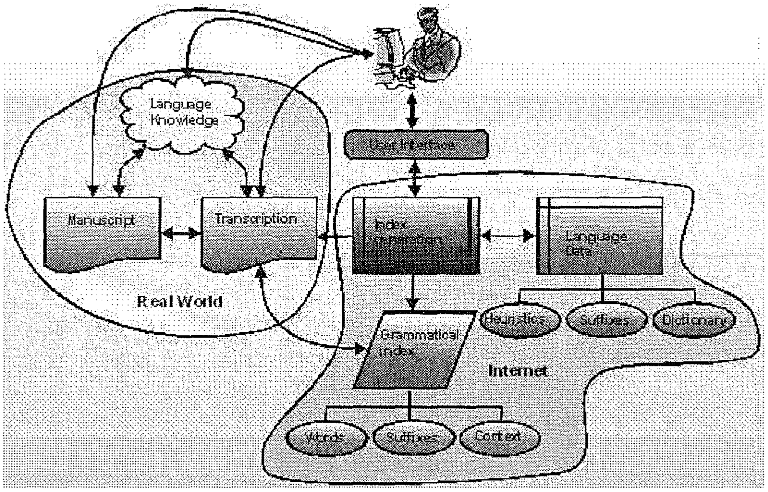


Figure 4

Figure 4 shows how the information in the transcription, the suffix lists, the dictionary and the knowledge of the user can be integrated to form a methodology. Information Technology not only reduces the time to produce grammatical indexes but also increases the amount of information that can be derived from them. The methodology above integrates both horizontally and vertically.

References

ARAT, R. Rahmeti, *Kutadgubilig III İndeks,* (haz. K. Eraslan, O. F. Sertkaya, N. Yüce) , Türk Kültürünü Araştırma Enstitüsü Yayınları: 47, İstanbul 1979.

www.concordancesoftware.co.uk

FLANAGAN, David, *JavaScript: The Definitive Guide 4th Edition,* O'Reilly ® 2002.

HAMILTON, J. R., *İyi ve Kötü Prens Öyküsü,* (çev. Vedat Köken), Türk Dil Kurumu Yayınları: 682, Ankara 1998.

IIS 5.1 Microsoft ® Active Server Pages (ASP).

Internet Explorer Copyright © 2006 Microsoft Corporation, *Version 7.0.*

Microsoft ® *Office XP 2002*, © Microsoft Corporation 1994-2001.

Microsoft ® *Word 2002*, Copyright © Microsoft Corporation 1983-2001.

Microsoft Visual Basic 6.3, Copyright © 1987-2001, Microsoft Corp.

TEKCAN, Münevver, " Hazret-i Meryem Kitabı", *İstanbul Üniversitesi Edebiyat Fakültesi Türk Dili ve Edebiyatı Dergisi 2004*, Vol. XXXII, İstanbul 2005, pp. 185-216.

TEKCAN, Münevver, "Grammatical Index Generation for Turkish Texts" *Journal of Turkish Studies:* (Türklük Araştırmaları Dergisi) Festscrift in Honor of Orhan Okay III/ Harvard USA, Vol. 30/III, 2006, pp. 181-208.

NAUR, Peter, "Revised Report on tne Algorithmic Language ALGOL 60.", *Communications on the ACM*, Vol. 3 o. 5 pp. 299-314, May 1960.

VATEC, Vorislamische Alttürkische Texte: Elektronisches Corpus (http://vatec2.fkidg1.uni-frankfurt.de/vatec.htm)

WEISSINGER, A. Keyton, *ASP: In a Nutshell: A Desktop Quick Reference 2nd Edition*, O'Reilly ® 2000.

www.writersworkbench.com