Toward a Spellchecker for Albanian Language

Betim Çiço[1] and Vasian Cepa[2]*

[1] Polytechnic University of Tirana, Albania
Email: bcico@icc.al.eu.org
[2] Technische Universität Darmstadt, Germany
Email: cepa@informatik.tu-darmstadt.de

Abstract. Nowadays an automatic spellchecker is a basic requirement to produce quality written text in every language. This paper addresses some of the issues encountered during developing such a tool for Albanian language. Albanian language uses a basic Latin alphabet with very few added characters. However, its linguistics features makes it different from any other language.

1 Introduction

A spellchecker[1] is a computer tool that helps to find and correct typing errors in electronic text. There are usually to kind of error sources that introduce errors in electronic text. The first source of errors are human related errors done during typing and the second source are machine errors that result when existing printed document are scanned and converted to text using OCR techniques. The spellchecker can be used to detect errors and offer suggestions for corrections. The error detection phase is easily to be implemented. A hash table can be used to keep the set of known words (considered correct) and to compare each word in text in O(1) time. If the set of known words is large, the hash table may contain more than one entry in its internal bags, practically slowing the search by forcing it to be linear for words with similar hashing. Different techniques can be used to optimize hash tables, like double-hashing etc. A spellchecker however, is usually used iteratively. This means that the search (detection and suggestion generation) can be done for the next unknown word, while the user is judging the previous unknown error. While error detection is nearly mechanical suggesting correction is language dependent. Language dependency interacts in most of the phases of a spellchecker including the way we save words in the dictionary (optimize dictionary storage). For example some languages allow us to store only roots of the words and generate grammar forms from them.

The spellchecker basically consists of a spell engine and a dictionary. The dictionary is the base of knowledge augmented with rules that are usually part of the spell engine. The spell engine uses the rules to identify a minimum set of words from the dictionary that match a given word. In case of an exact match the word is considered correct.

* The work was done when author was part of a DAAD partnership program in University of Siegen, Germany

[1] The right term for *spellchecker* in Albanian is *drejtshkrimor*. Thanks to Merkur Beqiri, SCShqip@hotmail.com

In any other case the automatic spell engine should suggest a minimum of similar or possible words based on the rules and on the dictionary.

Currently there exists no any previous solution for helping with spelling the text in the Albanian language. Ignoring the grammar context one could argue to use an existing spell engine with a dictionary[2]. For this reason at first two implementations were considered. A free product called ASPC [5], which already has dictionaries for many western European languages and ASPELL [4], an open source implementation of classic ISPELL UNIX tool. The results obtained from testing these tools with a dictionary of around 6000 Albanian words were not optimal. Even they could find basic words they failed to suggest the right words in most cases. While this can be verified by the reader by using the tools with an Albanian alphabet, statistical analysis of correction ratios cannot be calculated[3] because the tools do not offer an interface to support such statistical analysis. It was this analysis inconvenience of these tools that made us to implement testing module as an integrated part of our prototype, as we will describe later, so the testing results can be repeated. Since this is the first solution that addresses the peculiarities of Albanian language there no predecessor specific spell checker tools to compare with. We try to identifying the points that make Albanian language different. A solution for word based spelling for Albanian language is represented in detail. The resulting prototype tool to test the solutions is also described. It is called *AlbPad* and it is developed in Java 1.2 [14]. Analysis of the achieved results is found near the end of the paper.

2 Albanian Words are Different

The problems with existing spellchecker tools brought the need for testing new algorithms on how to improve the correction ratio. Many spellcheckers use inexact string matching algorithms to identify minimum matching strings for a given word. The implementations are often based on the concept of Levenstein distance [2] implemented with dynamic programming algorithms (the edit, inset and delete distances), or structures that facilitate this kind of inexact matching like ternary trees [7], or even statistical based methods like n-grams [8, 9].

A characteristic of all these methods is that they use the natural order of a language. This is the alphabetical order of the characters the language uses in the alphabet. The order is important in calculating inexact matching. Albanian letters can use almost directly the ASCII encoding but the order of letters will be like this: a (97), b (98), c (99), d (100), e (101), f (102)..., Ç (128), ç (135), ë (137), ... In Albanian however the order of the letters (the locale) is: a, b, c, ç, d, dh, e, ë, d, f, ..., A, B, C, Ç, D, DH, Ë .... This means that an algorithm for calculating word distance based on ASCII is therefore not directly applicable for Albanian because while the position difference between [a-b] is the same, the difference for [c-ç] is 38 instead of 1. The same problem shows up with other letters like [e-ë] or even with compound letters of Albanian language like [g-gj], [s-sh], [l-ll], which occupy a full position in the alphabet by themselves, that is do not

[2] In this direction goes also a commercial effort found at `http://www.geocities.com/merkurbeqiri/drejtshkrimori.html`

[3] Well, it can be done but the programming details are tricky to consider.

have two letters semantics but count as only one. So it seems we have to define a locale [16] for the Albanian language.

However, there is still another problem. The Albanian letter $\ddot{E}$ is not part of ASCII for cp450 or cp850[4]. A Unicode [11] representation is thus required to represent the letter $\ddot{E}$. However the Unicode representation cannot solve the problem of natural ordering of the letters in the Alphabet since these symbols are scattered over the code. This requires that an internal coded structure to represent Albanian words. The spelling engine, even it may not use directly this encoding, must offer a way to let such a encoding to be specified. None of the considered spell engines offers this possibility. This means that existing implementations are not directly applicable to Albanian, since words like [cuke-çukë] that are too far in ASCII distance for English language are on the other hand very near (similar) for Albanian language.

Another point to consider is the average length of the words used in implementations of Levenstein distance [2] algorithms which tends to vary depending on the language. For English this it is usually 3 4, whereas for Albanian this is 6 - 7 (see the section related to AlbPad testing). For practical considerations some constants of the existing algorithms should be changed. The implementation of coding used in AlbPad is based in the fact that the letters in Albanian are different from ASCII cp850. The natural alphabetical order supported by the encoding shown only for lowercase letters is:

a, b, c, [ç], d, [dh], e, [ë], f, g, [gj], . . . , l, [ll], . . . , n, [nj], . . . , r, [rr], s, [sh], t, [th], . . . , x, [xh], z, [zh].

The capital letters follow the same order. There is a cedilla ç, an umlaut ë and nine compound (2 - character) letters. (The capital compound letters are written like [Gj] in the beginning of a sentence with only the first character in uppercase.) Unicode offers codes for a few of Albanian compound letters like [Nj]. One cannot relay on that however, since they belong to different coding sets and it is difficult to keep track of different character codes.

3 A Metaphone Algorithm for Albanian

Some encoding schemes like Metaphone [1] and to some extend Soundex [12] algorithms try to reduce the set of diphthongs of a language into a much smaller set of characters. This is especially useful for languages like English, where there is generally not direct match between pronounced and spelled words. These algorithms can still be useful even for other languages like German or Albanian where there exists a more direct mapping of diphthongs into letters (group of characters). These algorithms can be useful to generate jumps in the search space, which are not possible by the first type of inexact string matching algorithms we saw before. Jumps can be useful to identify local optimal hills and thus can speed the search. This can be implemented with special coding of the Metaphone. The implementation in AlbPad uses a variation of this technique, based

[4] Other code pages are other that Latin set based ones cannot be used.

on an adaptation of the Metaphone algorithm as shown next combined with binary search.

The rules for reducing Albanian words are simpler than those for English due to the fact the Albanian diphthongs have direct letter matches into alphabet. Below is the full algorithm:

Double letters are removed (there are very few of such special words in Albanian). A vowel is kept only if it is the first letter of a word or if it is the second letter and first letter is *H* with these two exceptions:

1. *Ë* - if it is first letter it goes into A (if *H* is the first letter, then after it is still considered as the first letter of the word)
2. *Y* - in the beginning of the word goes to *I*

The other letters are converted based on this mapping table:

B -> B	K -> K	S -> S
C -> Q	L -> L	SH -> SH
Ç -> Q	LL -> L	T -> T
D -> T	M -> M	TH -> TH
DH -> TH	N -> N	V -> F
F -> F	NJ -> N	X -> S
G -> GJ	P -> P	XH -> GJ
GJ -> GJ	Q -> Q	Z -> S
H -> (if not after s, d, t) removed	R -> R	ZH -> SH
J -> J	RR -> R	

Table 1. Other Metaphone Mappings

We will use G for GJ, X for SH and Z for TH. Note that G < X < Z and that our Metaphone coding is ASCII ordered (apart from *Ë* which requires another coding). The Metaphone will have a maximal length of 6 characters. We consider here *S* as differed from the compound *SH*. The corresponding sounds are quite distinct in Albanian[5]. Thus, our reduced alphabet of becomes: vowels + BFGJKLMNPQRSXTZ.

Applying Metaphone over the dictionary can be seen as a one-way transformation. It results in a new dictionary whose entries group the entries of the original dictionary. The inexact search algorithms can be applied (with some modification) to this new dictionary, to do meta-searches over the original dictionary. AlbPad used this technique to identify similar word sets.

4 Heuristics and Grammar Considerations

Heuristics are based on well-known facts about the language or about its usage. For example, in Albanian, *asht* and *është* are similar since they are dialectic forms of the same word. Another one is the use of *e* instead of *ë* into typed words, since most

[5] This is different from Greek where these sounds are quite similar.

people type Albanian with non Albanian keyboards. While these facts are simple the collection of them and conversion of them into proper rules remains difficult. At the time of this writing, there are no published heuristics about Albanian. The implementation of AlbPad spellchecker uses a few of them, but there is still room for improvement here. Heuristics may optimize search by eliminating sets of wrong words.

Another heuristic approach used in AlbPad is related to the use of keyboard. When typing there is more chance that a mistyped letter has been wrongly replaced by one found near it in keyboard (we are assuming a QWERTY layout). A keyboard layout for Albanian exists on Microsoft Windows. We are not aware that it is an approved standard and to our opinion it has not a natural layout[6]. Thus at the moment AlbPad heuristics are valid only for a US keyboard layout. We use a generative approach by generating more that one related word for words not found in the dictionary, based on the standard layout and pass all of them to the spell engine. Depending on the results an intersection or union of suggestions sets is returned. ASPELL [4] uses a similar generation technique for common grammar forms of a given word.

In above discussion we simplified the scope of spell checking to that of finding suggestions for errors found in single words. This means we are dealing here only with a finite choice grammar [13] representation of the Albanian language, which of course cannot fully model all relations that exist in a natural language. The real grammar of the language is complex and has many exceptions. The existing studies [15] relay on a general description of it and rarely into an enumeration or rules. There are many reasons for this, but for the moment there are however no studies that may be used to produce a set of formal rules for any other type of grammar for subsets of Albanian language or for derived and compound words generation. It is also unclear if such efficient rules can ever be formed. Semantic checking remains to be reconsidered in the future when more language studies will be available. A single word based spellchecker is currently the only possibility.

There is however a practical suggestion[7] to associate dangling surnames with the prefixing article, which is tied in Albanian, for example: *i mire*, *e mire*, *i kuq*, *e kuqe*, could be treated and saved into dictionary as single words. This simple context possibility is not used in the current AlbPad implementation.

Being not able to generate forms of derived and compounds words from their roots, results also in a much larger dictionary to be saved. Not only word roots, but also derived forms and conjugations and declinations should be stored. Thus the size of dictionary can get large. In the AlbPad implementation we save all the forms of a given word into a dictionary. This is not a problem with this prototype given. It can handle this way up to 20000 words, but for larger dictionary lists, some optimization will be necessary.

The AlbPad spellchecker was tested with a dictionary of over 6000 Albanian words. The source of words was the free electronic Albanian text found in the Internet from several Albanian newspapers sites scanned during a week period[8].

[6] To us, it seems more like a quickly modified German keyboard.
[7] Suggested during an informal talk, by someone in the Albanian Academy of Sciences.
[8] Commercial electronic dictionary assemblies for Albanian are available even thought not so complete, e.g. from `http://www.ectaco.com`

The pages were automatically downloaded everyday and the resulting HTML pages were scanned with a special Tokenizer to grab the words. A set difference was taken for every list of words obtained this way (ignoring single letter words and words with numbers or with non Albanian characters in them) with the words already in dictionary. The resulting ordered lists were briefly examined by eye to remove any errors. We think many of these newspapers use no spellchecker which results in errors in the words gathered. The resulting dictionary should not be considered 100% correct. However it can serve as a basis for testing other texts against it. The newspapers below were used as primary text sources:

- www.shekulli.com.al (The words' spellings were mostly correct.)
- www.balkanweb.com (They may use an Italian spellchecker with an Albanian dictionary[9]. The results were strange sometimes.)
- www.kohajone.com (Most of the errors found in the dictionary file come from this site. Especially they never used the letter *ë*.)
- www.zeriipopullit.com (The spelling was quite good here.)

The comments given inside the braces were valid at the time of testing[10].

5 Developed Details and Tools

The prototype uses a mixture of ways to analyst and produce error correction suggestion sets based on word distance and metaphone word similarity. The word dictionary is kept sorted. The first time the spell engine accesses the dictionary a hash table and an ordered metaphone indexes are created. For dictionaries with less that 20 thousand words there is only a small visible overhead for these operations. For bigger dictionaries the hashtable index and the metaphone should be precomputed and saved as an index file, which is not done in this prototype. The hashtable index is used to test for set membership in O(1) time. This way we can find quickly if a word is correct or not. If the word is not correct a set of possible error variations are generated, based on keyboard heuristics as described above and a few language heuristics. For each entity of these possible errors list a search is done by the spell engine, taking as final result the intersection of the individual suggestions sets of each entity. The extension of error entity based on heuristics is used in spell checker solutions like ASPELL [4] to generate derived English word forms based on composite or derived word formation rules. Given the lack of such well-formed rules for Albanian here we use only the heuristics we mentioned. The suggestion set for a single entity can be computed by iterating over all the list of dictionary words and ordering it based on Levenstein distance calculation. This naive approach can take a long time to calculate even if some parts of dynamic programming tables are changed when the number of the words in dictionary is more than a few thousand. Our approach here is to localize the search and perform distance calculations in the localized sets. The algorithm details are given next with explanation details that follow:

[9] Even there are some speculations about the misuse of language from unqualified journalists.
[10] That is, February 2001.

1. compute the entry metaphone
2. search the metaphone index for the nearest (ν) alphabetic metaphone using binary search
3. take a +/-δ interval of metaphone entries around the nearest one ν found
4. use Levenstein distance calculation over metaphone elements in the (ν - δ, ν + δ) to eliminate the elements that are farer that a given threshold θ_1
5. use Levenstein distance calculation the corresponding dictionary words mapped from the metaphone list, ordering the results based on the distance calculation
6. drop the words with distance bigger than a given threshold θ_2

The step 2 localized the possible word set in O(log(n)) time, but has a drawback that the result set is more accurate when the error is in end of the word, rather than in the beginning. The heuristic possible errors entity generator tries to compensate this by generating entries with different possible beginnings.
The δ parameter is step 3, and θ_1 and θ_2 in steps 4, 6 are a linear function of the size of the dictionary. The function constants for these parameters can be found and optimized by trial and error.
The binary search and the Levenstein distance calculations are done over the Albanian alphabet taking into consideration natural letter ordering and compound letters.

The suggestion set found this was for each possible generated error entry is merged. The possible generated errors are first ordered using Levenstein distance calculation via original error. Then the suggestion set entries are ordered in rotating order for each entry. That is let the original error entry be ε_0 with result suggestion set $\Sigma = \{\sigma_{00}, \sigma_{01}, \ldots, \sigma_{0N_0}\}$, where N_0 is the number of suggested words set for this entry. The other M ordered possible generated error entries with found suggestion set results be of form $(\varepsilon_i; \Sigma = \{\sigma_{00}, \sigma_{01}, \ldots, \sigma_{0N_0}\})$, where i is the ordered index of the M other possible entries generated. Then the overall ordering is:

$$(\sigma_{00}, \sigma_{10}, \ldots, \sigma_{M0}, \sigma_{01}, \sigma_{11}, \ldots, \sigma_{M1}, \ldots, \sigma_{0N_0}, \sigma_{1N_1}, \ldots, \sigma_{MN_M})$$

Note that lists for entries may vary in length. An arbitrary threshold θ_3 interrupts the lists if it is too long. The user can specify a more conservative threshold via a 'Limit Result' check menu.

This optimized calculation still has a performance price. However the user interacts with the spell engine during the spell check of the text. So we use a producer-producer-consumer multi-thread architecture. The first producer sets all the text words one by one the spell engine. The spell engine if the word is in dictionary asks immediately for the next word (O(1) time). The error entries consume search time. While the first error is presented to the user via the dialog in Fig. 2, the first producer continues to check word to the spell engine and while the user (GUI consumer thread) decides what to do with the result set, the next result is computed. This way it seems to the user that the next error suggestion is shown almost immediately. A few tricks are needed to ensure the right positioning in the text buffer by the three threads, since the length of the text buffer changes as the user continues spelling, but they are of no direct interest to the discussion here. Also the spell engine changes as the spell progress, new words can be added to the dictionary and other words should be ignored if user decides so, during the entire search. A few other tricks maintain the spell index tables in correct state when such changes are needed.

To test the ideas a set of tools were implement under the name of AlbPad, an editor and spell checker for Albanian language. The development of AlbPad prototype and its related tools was done in Sun Java 2 [14] language and the source is Java 1.2 compatible. Some of the main tools developed are listed below. It is recommended to consult command line parameters and AlbPad documentation if you need more details than given here.

Dictionary Utilities: are used to facilitate the generation of dictionary and to manage it. The tools can also invoke Tokenizer to generate ordered lists of words from raw text.

```
java -cp albpad.jar com.vpcepa.spellText.DictUtils
```

The text statistics utility: is similar to wc tool in UNIX, but is Albanian language aware and reports some more statistics.

```
java -cp albpad.jar com.vpcepa.spell.WStat
```

A text based interface to the spell engine that was used mainly during testing, shown in Fig. 1:

```
java -cp albpad.jar com.vpcepa.spellText.Spell
```

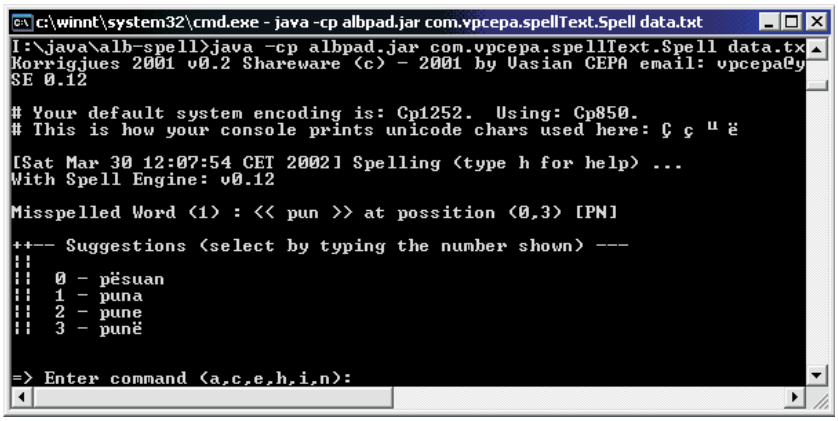

Fig. 1. The spell checker text based interface

The text interface gives in braces is the Metaphone coding of the word.
The graphical editor: The final interface is a full-featured text editor for Albanian. It can also emulate keyboard to type Albanian characters in two different user customized ways.

```
javaw -jar albpad.jar
```

A screen from spell checker sub-tool dialog in work is shown in Fig. 2:

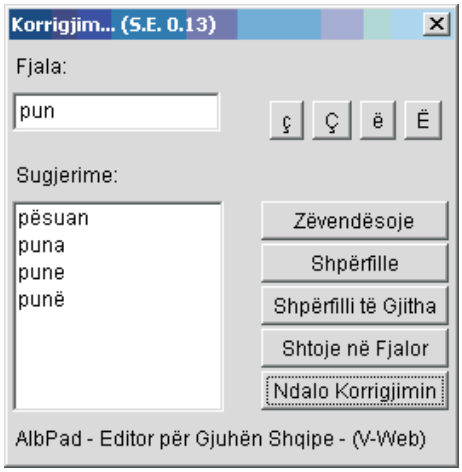

Fig. 2. AlbPad spell checker dialog box *(Replace, Ignore, Ignore All, Add to Dictionary, Stop Spelling)*

The current version of spell engine is 0.13. The relative small dictionary used does not allow a general usage of the spell engine. The set of developed is offered for evaluation and can be freely downloaded from [6]. Please check the documentation that comes with AlbPad for more features and usage details.

We also made testing a integrated feature of our prototype so it can be used by others to test its performance. Automatic testing of word-based spellcheckers can be done in following way. A list of words is taken from the dictionary. Errors are introduced at random in all the words of list. The errors can be single- or multiple-letter ones. The new list is passed to spell engine. For every misspelled word one can count if the right version is or not part of the answers returned and the number of entries returned. One must count also if there is an error in first letter of the word[11]. Various statistics can be built then from these numbers. The utility:

```
java cp albpad.jar java com.vpcepa.spell.STest
```

was developed based on these guidelines and used to gather statistics about the spell engine. The detailed results for a testing set of 1072 words with errors introduced with normal probability are given in the Fig. 4 in appendix, whereas the details about the testing word set are given in Table 2.

As we can see from the table the current spell engine can correct with success rate of 44% for words that contains only one error shown in Fig. 3. The success rate falls to 13% when there are two errors and to 4% when there are three errors[12].

[11] Sometimes first-letter errors are more difficult to be handled.

[12] Having three errors for word means that a three-letter word is now another word, unrelated to the first. This especially true for four-letter words with four introduced errors as can be seen from the graph.

Words in dictionary	6429
Words in every test set	1072
Average word length in characters	8
Average word length in Albanian letters	7

Table 2. Testing word set details

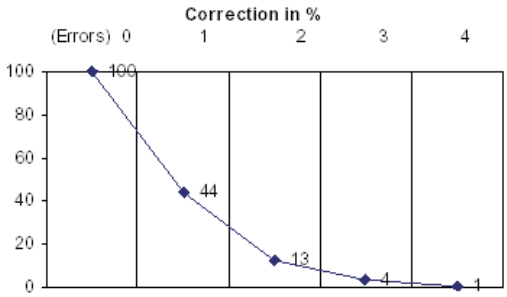

Fig. 3. Correction rate versus letter error count

Since the introduction of errors is uniformly distributed, it is expected that first letter error percentage be near {[1/(word length in characters)] * number of errors} * 100%. The number of errors is slightly bigger form the edit distance when error number is >1, since we allow a previously changed letter to be re-chosen for next changing.

During automatic generation of the words with errors some of the transformed words may be also in the dictionary (that is we obtain a new valid word). These are shown in the Exist column(s) of the table. The percentages of successes assume that existing words are also errors. The table shows also that the current implementation is not so effective when the error is found in the first letter of the word (this is generally a problem with all inexact string matching algorithms). The average suggestion set size is low (results are obtained with the *Limit suggested set* option of the spell engine) mainly because there are not so many words in the dictionary, thus there are not many options for every word to select from. For specific words with more form options there are more suggestions.

Since this is the first implementation of a spell checker in Albanian there are no other spell engines to compare with. It has not yet a quite good correction rate, but it can serve as a reference for other implementations in the future.

6 Conclusions

We think that the work done has achieved its goals as a starting point for a more elaborate implementation of a spell checker for Albanian. This implementation is based only on single words and not in any context sensitive rules for the language. A complete

solution would require cooperation with language experts to define rules and proper grammar descriptions for many semantic forms present in the language.

The realization was done by computer science experts communicating ideas with some Albanian language experts. Any future work would require the work of a group of experts of related fields of linguistics and computer science, to evident the rules and feature that are peculiar to Albanian and are not found in other languages for which a solution already exists.

References

1. L. Philips: Hanging on the Metaphone, *Computer Language*, 7(12):39-43 (1990)
2. G.A. Stephen: String Search, (1992), `http://www.ecafe.org/~graham/cv/`
3. P.A.V. Hall, G.R. Dowling: Approximate String Matching, *ACM Computing Surveys*, 12(4):381-402, (1980)
4. ASpell Tool: `http://sourceforge.net/projects/aspell/`
5. All-Purpose Spell Checker 4.0: `http://www.pakware.com/aspc/`
6. AlbPad Editor: `http://vpcepa.virtualave.net/software/spell/index.html`
7. J. Bentley, R Sedgewick: Ternary Search Trees, *Dr. Dobb's Journal*, (1998), `http://www.ddj.com/documents/s=921/ddj9804a/9804a.htm`
8. P. Sibun, J.C. Reynar: Language Identification: Examining the Issues, *Proceedings 5th Annual Conference on Document Analysis and Information Retrieval (SDAIR)*, Las Vegas NV, (1996), `http://www.cis.upenn.edu/~jcreynar/sdair96.ps.gz`
9. W. Kim: A Study on Statistical Language Identification, Ph.D. Qualifying project report, The Johns Hopkins Univ. (2001), `http://www.cs.jhu.edu/~woosung/ps/lid.ps`
10. R. Lutz, S. Greene: Measuring Phonological Similarity: the Case of Personal Names, `http://www.las-inc.com/nameinfo/wp_lsa.htm`
11. Unicode Character Chars Homepage: `http://www.unicode.org`
12. Soundex: `http://www.acorn.net/gen/soundex.html`
13. D. Grune, C.J. Jacobs: *Parsing Techniques*, Ellis Horwood Limited, (1990)
14. Sun Java Technology Homepage: `http://java.sun.com`
15. Drejtshkrimi i Gjuhës Shqipe: Tiranë (1973), Botim i A.SH.R.P.SH., Instituti i Gjuhësisë dhe Letërsisë
16. Code Pages: MSDN, `ms-help://MS.VSCC/MS.MSDNVS/vclib/html/_crt_code_pages.htm`

Appendix: Testing Results Table

Fig. 4 shows the testing results. Each test was done with a random set of 1072 words selected from the dictionary. See Table 2 for details. Each line in Fig. 4 represents such a test case. Table 3 gives more details about the meaning of the text if header row of Fig. 4.

						Error	Error %	Error	Error %	Size	Size when Found
1	1	0	491	46	46	161	15	4	2	1	2
	5	0	479	45	45	162	15	4	2	1	2
	8	1	472	44	44	159	15	5	3	1	2
	2	0	497	46	46	171	16	0	0	1	2
	10	1	457	43	43	170	16	5	3	1	2
	7	1	465	43	44	189	18	7	4	1	2
2	15	1	135	13	13	274	26	1	0	1	2
	7	1	142	13	13	274	26	3	1	1	2
	10	1	134	13	13	258	24	0	0	1	2
	9	1	141	13	13	294	27	0	0	1	2
	8	1	144	13	14	285	27	4	1	1	2
	9	1	132	12	12	284	27	3	1	1	2
3	3	0	47	4	4	385	36	2	1	0	2
	3	0	37	3	3	398	37	2	1	0	2
	4	0	38	4	4	401	37	2	0	0	2
	3	0	40	4	4	403	38	1	0	0	2
	5	0	50	5	5	415	39	1	0	0	2
	2	0	42	4	4	403	38	2	0	0	2
4	1	0	12	1	1	493	46	0	0	0	2
	1	0	14	1	1	490	46	0	0	0	2
	1	0	13	1	1	520	49	2	0	0	3
	0	0	14	1	1	502	47	0	0	0	2
	1	0	15	1	1	475	44	2	0	0	3
	3	0	16	1	1	488	46	3	1	0	2

Fig. 4. AlbPad spell checker testing results

Errors	- the number of [errors] introduced in each word (mutated)
Exists	- the mutated word is still a valid word and exist in the dictionary
Found	- the unchanged word that were [found] in the returned suggested sets
Found%	- (Found / 1072) * 100
Total Found%	- like Found% but here Exists words are calculated as success
First Letter Error	- mutated words where first letter has mutation (error)
First Letter Error%	- (First Letter Error / 1072) * 100
Found & First letter error	- as it says
Found & First letter error%	- as it says, but in percent over 1072
Avg. Suggestion Size	- the average size of the suggested result set
Avg. Suggestion Size when Found	- as it says

Table 3. Legend for Fig. 4