Advantages of online spellchecking: a Croatian example

Sandor Dembitz*,†, Mirko Randić and Gordan Gledec

Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia

SUMMARY

Online spellchecking is commonly regarded as an auxiliary way of performing spellchecking. However, it offers a unique opportunity to constantly improve spellchecker linguistic functionality through interaction with the community of spellchecker users. Such a possibility is crucial for spellchecking in non-central and under-resourced languages, in order to overcome gaps in NLP tools between them and central languages. The paper describes Hascheck, a Croatian online spellchecker able to learn words from texts it receives. It started as the first Croatian spellchecker, hence as a basic NLP tool for an under-resourced language, but due to its learning ability it demonstrates linguistic functionality comparable to that of conventional central-language spellcheckers. Based on these experiences we also discuss the future of online spellchecking in the context of global NLP tasks. Copyright © 2010 John Wiley & Sons, Ltd.

Received 9 November 2009; Revised 8 October 2010; Accepted 12 October 2010

KEY WORDS: language corpus; spellchecker traffic; supervised learning; mathematical modeling of spellchecking parameters; Heaps’ law; workload anticipation

1. INTRODUCTION

Online spellchecking is commonly regarded as an auxiliary way of performing spellchecking. When one says ‘spellchecker’, or ‘spell checker’ as it is also written, one usually refers to a tool embedded in the writing environment of a personal computer. When it is invoked by editing software or the user, a conventional spellchecker operates at token level, as the text is typed. If a token is not found in the spellchecker dictionary, or if it does not follow the capitalization rules also contained in the dictionary, the spellchecker reports an error, marking the token within the text. An error message is usually followed by correction suggestions. The suggestions are based on dictionary content and some additional data incorporated in the spell-checking program. If a token is a so-called false positive, i.e. a correct word gets marked as an error because it is not contained in the dictionary, most spellcheckers allow it to be added into the user-defined customized dictionary. Besides the interactive mode of operation in which errors are reported as they are typed, thus allowing editing on the fly, conventional spellcheckers may also operate in batch mode, checking the entire document submitted for proofreading and reporting whatever is not found in the dictionary, thus leaving the editing for the text post-processing. Many conventional spellcheckers let the user to choose the mode of operation.

An online spellchecker usually does not operate interactively. It checks a submitted document as a conventional spellchecker in batch mode does. Another disadvantage of online spellchecking is that it does not preserve the original document formatting. The document is usually converted

*Correspondence to: Sandor Dembitz, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia.
†E-mail: sandor.dembitz@fer.hr

Copyright © 2010 John Wiley & Sons, Ltd.
The basic advantage of online spellchecking lies in a community of users gathered around an online spellchecker. If the community is large enough, it may be considered as a representative sample of people that use the language and writing system treated by the spellchecker. Unlike conventional spellchecking, where the user-tool relation is one-to-one, in online spellchecking we have a many-to-one relation. The new relation is potentially very productive for spellchecker dictionary updating. Unlike the user-defined customized dictionary, which stores the false positives of only one user, all the false positives that occur in online spellchecking can be stored in a single dictionary and used by the entire community. Hence, an online spellchecker offers the possibility of constantly improving its linguistic functionality through interaction with the community of users.

The dictionary is a crucial part of any spellchecker. There are three main approaches to its construction:

1. **Word-list compilation**
2. Compiling a **word (.dic)** file and an **affix (.aff)** file separately [1–6]
3. Describing possible word-forms by two-level morphology [7–9]

The latter two approaches are dominant in conventional spellchecking. Word-list compilation is generally considered a ‘primitive’ and memory-consuming way of creating a spellchecker dictionary. However, in the case of online spellchecking memory limits are not as important as they would be for a conventional spellchecker. The online spellchecker dictionary is resident (in one main copy) on a server which offers spellchecking service. Even a word list of 100 000 000 entities (word-types) can be stored, maintained and handled by a modest server. Such a dictionary could consume (non-compressed) approximately 1 GB of disk space. A word-list dictionary, i.e. a simple file containing all word-types that were found in a language, can easily be updated with new words found in the course of online spellchecking. It is important to note that no dictionary can contain all word-types that may occur in a language, since a language stays alive only if it is able to produce or borrow new words, with the word forms belonging to them, hence new word-types. One advantage of the word-list compilation approach is the simplicity of the spellchecking engine. The main advantage lies in the possibility to use open-source approximate string-matching software [10, 11]. The software allows easy programming of Metaphone-like programs [12, 13] for spellchecking purposes.

Spellchecking is a privilege of approximately 100 languages (writing systems) of the world [14]. This is a small number, considering that more than 1000 languages (writing systems) appear on the World Wide Web (WWW), a number which is increasing every day. Hence, approximately 90% of world languages that use the written form of communication do not have a basic natural language processing (NLP) tool, a spellchecker. These languages have to be considered as under-resourced in terms of NLP systems. The problem of the under-resourced languages has been recognized, and some effort has been made to improve the situation. In the *An Crúbadán* project, dictionaries for open-source spellchecking engines were compiled for a dozen under-resourced languages, and another dozen new dictionaries have been announced to come out soon [15]. The XNLRDF (Open Source Natural Language Resource Description Framework) project offered online spellchecking for more than 1000 under-resourced languages, but being ‘an unfunded garage project’ [16, p. 207], it has always suffered from poor accessibility.

Languages with established NLP systems may be divided into ‘central’ and ‘non-central’ languages [17]. The central languages consume the most research funding in the NLP domain. They are also characterized by competition between various R&D teams and NLP software producing companies, which results in a number of very sophisticated NLP tools appearing on the market almost on a daily basis, including sophisticated machine translation systems, even speech-to-speech translation systems. In contrast, non-central languages are generally characterized by lower NLP system sophistication due to the lack of research funding and competition in the R&D and software-producing sector.
Many non-central languages demonstrate NLP deficiencies starting with spellchecking. Their spellcheckers often have lower linguistic functionality compared with those of central languages. The lower spellchecker linguistic functionality is seen in low spellchecker recall and low suggestion adequacy. The third functionality parameter, the precision or the degree to which the checker rejects all invalid words, may be high, but precision does not contribute much to linguistic functionality if the recall, or the degree to which the checker accepts all valid words, is low, because each spellchecker user becomes distracted by highlighted false positives (non-recognized valid words) mixed together with invalid words. Because of that many genuine misspellings and typos may pass undetected and uncorrected.

Standards for evaluation of NLP tools, including spellcheckers, were developed by the TEMAA Project (http://www.cst.dk/temaa/D16/d16exp.html, also see [18]). Instead of recall, one may use text coverage, a parameter that tells how many tokens the spellchecker recognizes on average in a real text or corpus. Let us illustrate the text coverage–precision relation with numbers. In under-resourced languages it is normal for the first or only spellchecker to recognize only 85% of all valid tokens in a text. In non-central languages text coverage of 95% is considered good. Only in a few central languages may one expect a spellchecker with text coverage of 98%. Since the portion of misspellings and typos in a text, according to our experience based on processing of a huge number of texts (Section 3, Figure 8), rarely exceeds 2–3%, this means that only users in central languages have the privilege of a good ratio between false positives and misspelling/typo detection when doing spellchecking. All others have some trouble in distinguishing what is a genuine misspelling or typo, and what is a correctly written word not recognized by the spellchecker. Finally, if text coverage is low, it is obvious that suggestion adequacy, i.e. whether or not the checker provides correct suggestions for non-recognized tokens, cannot be high. Such a spellchecker will suggest inadequate correction(s) for many false positives and this may only inconvenience users.

We believe that online spellchecking, with distributed and collective nature of the project, is an effective way to overcome the functionality gap between central and non-central or under-resourced languages. Our opinion will be illustrated with experiences gained through developing, maintaining and operating Hascheck, the Croatian academic spellchecker—the initial letter H comes from its original full name (Croatian=Hrvatski). It started as an online spellchecker for the under-resourced Croatian language (Hascheck is the first Croatian spellchecker), went through the middle phase (although Croatian is still a non-central language in many NLP aspects), and nowadays demonstrates linguistic functionality equal to those of conventional spellcheckers for central languages.

Section 2 gives a description of Hascheck and the conditions under which it was developed. In Section 3 we describe Hascheck traffic in general and introduce its basic parameters, including a learning index (LI), which differentiates Hascheck from all other spellcheckers. This section also provides mathematical models of traffic growth. Section 4 exploits the abundance of traffic data for mathematical modeling of the behavior of spellchecking parameters in time. This section also introduces Heaps’ law applied to Hascheck as a tool for predicting learning volume and workload caused by learning. In Section 5 we consider the future of online spellchecking in general, while Section 6 brings our concluding remarks.

2. REMARKS ON THE CROATIAN LANGUAGE AND HASCHECK ARCHITECTURE

Croatia is a West Balkan country with approximately 4.5 million inhabitants. It borders Slovenia, Hungary, Serbia, Montenegro, Bosnia & Herzegovina and Italy (across the Adriatic Sea). The Croatian language is the official (state) language in Croatia. It belongs to the family of Slavic languages. The Croatian writing system is based on Latin script; Croatian orthography is (mostly) phonetically based. It uses five letters with diacritics to express Croatian sounds: ‘ć’ corresponds to the English digraph ‘ch’ in ‘checker’, ‘ć’ corresponds to the Dutch digraph ‘tj’ in ‘Aantjes’, ‘dˇ’ corresponds to the Italian digraph ‘gi’ in ‘Giulia’, ‘š’ corresponds to the English digraph ‘sh’ in ‘shop’, and ‘ž’ corresponds to the French letter ‘j’ in ‘Jacques’. In the Croatian alphabet three digraphs are treated as letters because they correspond to single sounds: ‘dž’ corresponds
to the English letter ‘j’ in ‘job’, ‘nj’ corresponds to the Spanish letter ‘ñ’ in ‘Buñuel’, and ‘lj’
corresponds to the Italian digraph for palatal L ‘gl’ in ‘Ventimiglia’. In total, the Croatian alphabet
contains 30 letters, five vowels and 25 consonants. Besides five open vowels (a, e, i, o, u) the
Croatian soundsystem uses two diphthongs, short and long /je/. In writing they are represented
by the digraph ‘je’ (short) and trigraph ‘ije’ (long), respectively. The Croatian writing system
generally does not use accents. Foreign names borrowed from languages with Latin script are
written in Croatian in their original orthography. Names from languages with non-Latin script
should be transliterated according to Croatian transliteration rules, but in contemporary Croatian
writing practice English transliteration of names is very often used; it makes them recognizable
because of their frequent appearance in that form in international use.

The language is still in a process of standardization. Three orthography standards, which differ
in many details, are in use in Croatia nowadays [19–21]. This is so because for a century or more
Croatian was treated as a ‘variant’ of the Serbo-Croatian language, one of the official languages
(the most used) in former Yugoslavia. After the former state split apart, the Croatian language
entered a dynamic phase of ‘emancipation’. This resulted in the instability mentioned above. It is
important to emphasize that the orthographies mentioned above are not competing in the sense that
the Bokmål and Nynorsk standards are in Norwegian, for example. They only recommend different
ways of writing for particular classes of words like ‘greška/griješka’ (‘error’ in Croatian), where
the first solution is strictly phonetically based, whereas the second solution is morphologically
motivated, preserving the diphthong /je/ in writing because of its existence in the verb ‘griješiti’
(to make an error), from which the noun is derived.

The Croatian language is a highly inflected language, i.e. a language with many word forms
(word-types) of the same word. The first reliable estimate of dictionary size (in word-types)
needed to solve the spell-checking problem well in a Slavic language was for Russian [22]. This
estimate, giving a bottom border of 1 000 000 and top border of 100 000 000 word-types needed to
develop a Russian spellchecker, was in the Croatian case, although theoretically valid, in practice
unacceptable. A market of a few hundred thousand potential consumers does not accept approaches
appropriate for a hundred-million consumer market. Hence, the inflection problem had to be solved
in another way.

Starting from Croatian specific issues, the spellchecker for Croatia had to be:

• orthographically tolerant (meaning that the spellchecker should accept words from all three
  orthographies) and user-oriented (meaning that the acceptance depends on word variants
  appearing in writing);
• developable with modest manpower resources, likely with zero funding;
• economic in use of computational resources (disk-space, etc.);
• able to cope with Croatian inflections in a way acceptable to users;
• flexible regarding improvements in its linguistic functionality and easy to maintain.

These demands made Hascheck an intelligent system (Figure 1).

Hascheck demonstrates its intelligence in several ways:

• by using different colors to mark tokens that might be valid Croatian word-types and those
  that could not be (a novel concept carried out through fuzzy evaluation of unrecognized
  alphabetic strings in the Classifier block);
• by pointing out tokens which might be valid inflected forms of words or names contained
  in its dictionary (morphological tagging based on language statistics realized in the Guesser
  block);
• by offering the most probable corrections for non-tagged tokens (based on keyboard layout
  and language statistics; this function is realized in the Corrector block);
• by learning new word- and name-types from texts received for processing (expert post-
  processing supervised by humans, realized in the Learning System block).

Hascheck has two subsystems: the real-time subsystem, which reacts immediately to text
received for spellchecking (as many texts as needed can be processed at the same time), and the
Post-processing Subsystem, which uses collected process data and performs learning on a maintainer’s demand (Figure 1). The learning results in dictionary updating, i.e. in improving spellchecker functionality.

The post-processing subsystem is the part that makes Hascheck different from conventional online spellcheckers. It offers advantages which will be described in the following two subsections.

2.1. Real-time subsystem

In the Extractor block (Figure 1) tokens found in the dictionary are excluded from further processing. In conventional spellchecking, unrecognized tokens are passed on to the correction suggesting subprogram directly, but in the case of Hascheck two processes are performed before that.

Unrecognized tokens pass through the Classifier first. Our string-classifying algorithm is language-independent. It is based on a string-weight concept [23], where the weight is a measure derived from language statistics of variable order. Hascheck uses three statistical files for calculating string weight: binary statistics of non-positional 3-graphs, 4-graphs and 5-graphs that occur in Croatian words. An n-graph consists of n consecutive graphemes including blanks at the beginning and at the end of a word, respectively. The n-graph statistics for Classifier purposes are derived exclusively from words which obey Croatian orthographic rules, hence from what we call common Croatian words (see below). It is a statistical representation of the Croatian writing system. The original string-weight concept was much more complex, being an extension of the peculiarity calculation applied in the early UNIX program typo [24], with the difference that the measure for peculiarity was not drawn from the processed text but from the dictionary and the frequency of words occurring in it. We decided to simplify it because of the robustness of binary statistics. We preserved the term ‘peculiarity’ and we obtained four peculiarity classes for alphabetic strings not contained in Hascheck’s dictionary:

- extremely peculiar strings, containing at least one unknown 3-graph, marked in red in the Report (Figure 1);
- very peculiar strings, containing at least one unknown 4-graph, marked in brown;
- moderately peculiar strings, containing at least one unknown 5-graph, marked in blue;
- almost non-peculiar strings, i.e. strings with all known 5-graphs, marked in green.
In the latest Hascheck interface, very peculiar and moderately peculiar strings were reduced to the same peculiarity class and both types of strings became blue. Short alphabetic strings with three or fewer letters, on which \( n \)-graph statistics cannot be applied, also obtain their color, as well as strings built of letters and figures, like Number2, or strings with an unusual combination of capital and small letters, like REport.

At the beginning of Hascheck’s life, when its text coverage was low and many valid Croatian words were reported as unknown, the peculiarity concept was demonstrated to be of great practical value. Approximately 80% of unrecognized valid Croatian words were classified as almost non-peculiar, and the rest of them were mostly classified as moderately peculiar. In contrast, misspellings and typos were classified in higher peculiarity classes, with a predominance of the highest peculiarity class (extremely peculiar strings). At that time Hascheck used to send its report in the form of an ordered list, and because of the classification selectivity it was possible to put most misspellings and typos at the top of the list, while valid but dictionary non-confirmed words were placed at the end. An explanation of the high selectivity lies in the phonetic character of Croatian orthography. Sound economy, occurring in all natural languages, in the Croatian case is turned into high predictability of letter sequences. We tested our classification method with English words too, but we never succeeded in putting more than 50% of unknown English words into the lowest peculiarity class.

Hascheck started operating with the Extractor and Classifier block only. The Guesser and Corrector block were introduced later, and their development and refinement depended very much on the size of the dictionary, which was regularly updated by the Learning System. It is worth knowing that Learning System automation largely depends on the state-of-the-art reached by the Guesser or the Corrector block. At the very beginning the Learning System could rely on classification results only.

The dictionary is central and the most valuable part of the system. It is organized in three word-list files:

- Word-Type (WT) file,
- Name-Type (NT) file,
- English-Type (EngT) file.

The initial WT-file was derived from the Croatian side of the English–Croatian Lexicographic Corpus (ECLC), a bilingual one-million-token corpus of approximately 80,000 elements (translation pairs) \([25]\). Besides the conventional bilingual dictionary elements—entry word and its translation(s)—ECLC contains a significant number of idiomatic elements, where the word usage is given by examples. The idiomatic elements were important for our choice; they gave us Croatian words in their inflected forms. The Croatian side of ECLC gave us a representative Croatian word-type sample, which, after careful and tedious manual spellchecking, produced the initial WT-file with approximately 100,000 different Croatian common word-types; by ‘common word’ we mean words which may occur written in small letters only, with an initial capital letter (at the start of a sentence, for example), or written in capital letters only, and which have not been borrowed with their orthography from foreign languages but belong intrinsically to the Croatian language itself. As mentioned before, only WT-file elements enter into the process of deriving \( n \)-graph statistics used by the Classifier.

The English side of ECLC was used to obtain a representative English word-type sample. The sample was checked with several English spellcheckers. This procedure gave us a deeper insight into the performance and linguistic functionality of these tools \([26]\). It was valuable information for further development. The result of spellchecking was merged with UNIX’s \texttt{spell} word list \([3]\) and with an old UNIVAC spellchecker word list \([27]\). After exclusion of words equally written in Croatian and English, like ‘atom’ or ‘zebra’ (approximately 1000 of them contained only in the WT-file), this produced our EngT-file with approximately 70,000 different English word-types. Our decision to include English in Croatian spellchecking is based on practice and intuition. English, as the modern \textit{lingua franca}, often comes mixed with Croatian in contemporary Croatian writing. We believe that something similar happens in many other languages too, probably with different intensities. The intuitional dimension of our decision can be supported today by the content profile.
of the Hascheck corpus. It is built of the following constituents:

- 89.4% of tokens belong to the WT-file (common Croatian words);
- 5.9% of tokens belong to the NT-file (names and the like);
- 1.7% of tokens are misspellings or typos;
- 1.6% of tokens belong to the EngT-file (common English words);
- 1.4% of tokens are purely numerical tokens, which are not subject to spellchecking.

Isolated English words or short phrases appear very often inside the Croatian sentences; one reason is a lack of adequate translations. We do not know any language-recognition software which functions well on the isolated word or short-phrase level, and therefore our early decision to include English in Croatian spellchecking has an a posteriori justification. The content profile tells without doubt that without the EngT-file it would be impossible to approach text-coverage of 98% or higher. Finally, because of the great distance between the two languages, in terms of Damerau–Levenshtein metrics [28, 29], no great fear exists that potential Croatian misspellings/typos could be treated as English, or vice versa [30].

The NT-file contains all case-sensitive elements of writing. These are: proper and other names, abbreviations, acronyms, as well as names with unusual use of small and capital letters, like LaTeX or WordPerfect, etc. Furthermore, it contains words from German, Hungarian, Italian and other foreign languages that appear in Croatian writing in their original orthography. The file started as an empty file, but due to learning it has increased by now to approximately 560 000 different name-types. At the same time the WT-file increased from the initial 100 000 to approximately 800 000 different Croatian common word-types. Only the EngT-file did not change its size, because all new English words that appeared in processed texts were placed into the NT-file.

There is a fourth file that may also be considered as a part of Hascheck’s dictionary. It is the so-called Error-Type file (ErrT-file) that contains all strings that have ever passed though Hascheck, but that have never been learned as common word-types or name-types, respectively. The file contains approximately 1.2 million different error-types. The ErrT-file is used to decide whether a token should pass through the Guesser block or not. If a token is in the ErrT-file, it goes directly to the Corrector block. Furthermore, the ErrT-file has been demonstrated to be very useful for accelerating learning speed.

It is the variety of word endings that makes a language highly inflected. However, the endings are predictable, not only morphologically but also statistically [31]. Considering the Croatian language, we implemented a statistical approach very early. Based on statistics of endings found in the WT- and NT-file, we have gradually developed our Guesser (Figure 1). The Guesser is able to tag various Croatian word endings with the content of the WT- or the NT-file. A successfully tagged token is considered as a correct but unknown word- or name-type, respectively. The file contains approximately 1.2 million different error-types. The ErrT-file is used to decide whether a token should pass through the Guesser block or not. If a token is in the ErrT-file, the tagging disregarded and the token goes to the Corrector block. For reasons of precision, successful tagging with WT-file content is considered as correct only if the token belongs to the lowest peculiarity class. Successful tagging with NT- and even EngT-file is considered as correct regardless of peculiarity class. (Some English words appear in Croatian in their inflected form; for example, ‘fitnessa’ as the Croatian genitive of the English noun ‘fitness’, appears on more than 15 000 Croatian WWW pages.)

Hascheck’s guessing precision is equal to 91.3% for word-types, and 87.6% for name-types. Guessing errors occur when an improper but allowable ending follows a proper root, which happens in writing, or when a root for an ending is misrecognized. An example: ‘marseillski’ will be tagged as a possible adjective derived from the French name ‘Marseille’, although the correct Croatian spelling, aside from the phonetic spelling ‘marejski’, is ‘marseilleski’, where the whole French root is preserved. It would be too tedious to treat all such exceptions, because, in general, the Croatian suffix ‘−ski’ discards the noun-ending vowel in the adjectival derivation. The Guesser should know that the French non-pronounced (unaccented) ending ‘−e’ must be preserved as root content, but for the whole NT-file content this would demand too much programming.

Hascheck’s guessing recall differs significantly between word- and name-types. While 81.9% of all common Croatian word-types ever learned were correctly tagged before learning, only 38.7% of name-types learned were correctly tagged before learning. Name occurrence in writing is very
random and unpredictable, while common words’ occurrence and their word forms are regulated by
language grammar; hence, they are much more predictable. This explains the differences in recall.

Generally speaking, the guessing could be extended to prefixes too. This could increase the
guessing recall for common word-types significantly. However, we feared reducing the precision
of guessing, since many productive Croatian prefixes are semantically motivated, while suffixes
are generally only grammatically motivated, so we have never done it in the real system. We did
several tests, but they only confirmed our fear.

At the end of the spell-checking procedure comes the Corrector block. This block was developed
gradually. At the very beginning only typical Croatian misspellings, the confusion of ‘ć’ and ‘č’
and incorrect diphthong presentation in writing (confusion of ‘ije’ and ‘je’) were communicated
with suggestions for error correction. When Hascheck’s text coverage stabilized at a level of 95%
and better, we introduced general error correction. For approximate string matching, we use the
freeware tool nrgrep [32]. The general error correction started with errors of edit distance 1 (one
letter substitution, one letter insertion, one letter deletion or transposition of two adjacent letters),
but with the increase in dictionary size it was extended to errors of edit distance 2. Because of the
phonetic character of Croatian writing, higher edit distance is not needed. It is worth noting that
the Croatian spellchecker for Microsoft writing tools offers corrections of edit distance 1 only.
Edit distance 2 also covers the majority of English misspellings and typos [33].

The results of approximate string matching are weighted according to keyboard layout and
some Croatian phonetic rules, but also by taking into account the frequency of words found in the
Hascheck corpus. The highest weight (equal to 10) is assigned to the typical Croatian misspellings
(interchanging ‘ć’ and ‘č’, and interchanging ‘ije’ and ‘je’). The same weight is assigned to a
broad class of letter substitutions caused by the phonetic character of the Croatian writing system,
like changing ‘d’ into ‘t’ before ‘k’, as happens in ‘sladak/slatka’, where both words mean ‘sweet
adj.’ but the first one is the nominative adjective for masculine gender, while the second one
is the nominative for feminine gender. This means that the correction candidate ‘slatka’ for the
misspelling ‘sladka’ will be weighted with 10. Weight equal to 5 is assigned to substitutions of
Croatian letters containing diacritics (ˇc, ´c, Y, ˇsa and ˇz) with their letter pairs without diacritics (c,
d, s and z), because of frequent occurrences of such errors in Croatian writing practice. Weight
equal to 3 is assigned to transposition of two adjacent letters, and to a letter substituted for a
neighboring keyboard letter, for example, substitution of ‘s’ for ‘a’. Other single substitutions,
as well as a single letter insertion or a single letter deletion, with the exception of the ‘ije/je’
case, are weighted with 1. These weights are multiplied by candidate word frequency from the
Hascheck frequency profile. The result gives an order in which the correction suggestions will be
presented to users, when candidates of edit distance 1 are considered. Only if there is no candidate
of edit distance 1, are candidates of edit distance 2 searched, but they are ranked according to their
frequencies only. If the number of correction candidates exceeds 10, all candidates whose final
weight is equal to or less than 1/10 000 of the first-ranked candidate’s weight are not displayed at
all. Nrgrep operates with all three dictionary files, but no phonetic rule using the weight of 10 is
applied when weighting correction candidates found in the NT- or EngT-files.

2.2. Post-processing subsystem

Each real-time process (spellchecking) is followed by a statistical record, which contains the
address of the text sender, date and time when a text was received for processing, data on text size
expressed in number of tokens, as well as number of unrecognized tokens and their distribution
over peculiarity classes. Furthermore, all unrecognized tokens with marks of peculiarity class are
recorded under the user’s address-date-time header. These are two basic files stored in the Process
Statistics & Log Files block (Figure 1). The third file contains frequency of recognized tokens,
which is used for dictionary statistic updating. A temporary log file containing ±5 token context
(if such exists) for each unrecognized token is also created. The file is deleted after each learning
process. Other statistical and log files are produced by the Learning System.

The Learning System is a part that crucially differs online from conventional spellchecking.
Because of the Learning System user language competence becomes a collective value.

DOI: 10.1002/spe
Figure 2. Sample prepared for learning.

The Learning System is activated when sufficient data are collected. This means that the Learning System must remember when the last learning was performed. Because of the fact that each processing has a unique address–date–time header in the log files, it is a trivial programming task. Unrecognized tokens collected between the last learning and last real-time processing are treated as elements of the language corpus. In such a form they once again pass through all Real-time Subsystem blocks. This time no exclusion of tokens existing in the ErrT-file is done, and all tokens, regardless of their peculiarity class, pass through the Guesser block. The result of the process is presented in Figure 2.

The record structure in Figure 2 is the following: first comes the peculiarity class mark (mark ‘−ss−’ means almost non-peculiar string), then the string and a number which says how many times the string occurred in the corpus, and finally either the tagging mark (‘P!’ means that the string is successfully tagged by the NT- or EngT-file, whereas ‘p!’ means that the string is successfully tagged by the WT-file) or suggestions for correction.

A look at the corresponding context file has confirmed that tagging has been correctly done in three cases in Figure 2. Lines 1 and 2 refer to Carlos Santana, a world-famous musician, and both tokens are inflected adjectives derived from his family name, while line 15 is the inflected adjective derived from the Croatian common word ‘astronaut’, one of the rare words that are spelled the same in English and Croatian. A further look at the context file has confirmed that the corrections in lines 5, 7, 9, 11, 12, 13, 14 and 16 have been well suggested. Misspelling ‘Sprem’ (line 5) refers to Karolina Šprem, a Croatian tennis player, ‘Vartekstovog’ (line 9) is a typo of an inflected adjective derived from the proper name Varteks (Croatian clothing factory), ‘Vidovic’ (line 11) refers to Petra Vidović, a Croatian celebrity, ‘amerikanci’ (line 14) refers to Americans (‘Amerikanci’ in Croatian) and ‘audi’ (line 16) refers to the Audi car brand. Lines 7, 12 and 13 are typical Croatian misspellings with the –ije-/je- and –č/-ć- problems.

A part where human intervention is needed comes with the following. Line 3 refers to Sonia Gandhi, the Indian politician, where ‘Sonije’ is the correct Croatian genitive of the foreign personal name ‘Sonia’, but very close to the genitive of a frequent Croatian name ‘Sonja’, which is given as the first-ranked correction. It is obvious that the decision ‘to learn or not’ is not trivial; frequency of occurrence may influence it. ‘Spiroskog’ (line 4) is the genitive of the Macedonian family name Spiroski, and there is no problem in learning it. ‘Suveretu’ (line 6) is the locative of the Italian municipality Suvereto, Livorno Province, Tuscany, and there is also no problem in learning it. Line 8 refers to the frequent Dutch family name Van der Valk, where ‘Valka’ is the Croatian genitive of ‘Valk’. However, ‘Valk’ is very close to the frequent Croatian family name ‘Vlak’, whose genitive is suggested as the first-ranked correction. Again the decision to learn or not must rely on human language competence. Finally, line 10 refers to the Vatan—Saint-Fargeau stage of the Tour de France 2009, where ‘Vatana’ is the genitive of ‘Vatan’, and there is no problem in learning it. These examples explain why learning should be supervised by humans. Some decisions cannot rely on machine intelligence; they must be made based on human capabilities and language.
competence. It is impossible to establish a metric able to estimate for each candidate word the benefit or risk of adding it into the dictionary. Two candidates, discussed as problematic above (‘Sonije’ and ‘Valka’), had never appeared as misspellings in the Hascheck corpus before their first appearance, and because of that they are ‘virgin’ candidates for learning. However, Croatian language competence tells us that they are potential misspellings of two frequent Croatian names. Including them is a risk for the future, without any confirmation in the past. A metric for the benefit of learning them could be established only if there were a program able to tell us how often ‘Sonia’ or ‘Van der Valk’ are going to be a subject of Croatian writing in the future. Such a program does not exist. Therefore, only an intuitional weighting of potential risk and possible benefit can be a criterion for learning them or not. Based on intuition it has been decided to learn ‘Sonije’, but not to learn ‘Valka’. Later processing confirmed that the intuitive decision made in this case was the correct one. Today we have four inflected forms of ‘Sonia’ in our NT-file. These are ‘Sonije’, ‘Soniji’, ‘Sonijom’ and ‘Soniju’, and their learning was contextually justified in all occurrences; they were never typos of the corresponding inflected forms of ‘Sonja’. Unlike ‘Sonia’, ‘Van der Valk’ never appeared again in any form in Croatian texts processed after its first appearance. As the examples indicate, intuition is applied mostly to name-type learning, while in cases of common Croatian word-type learning matching of a candidate with a frequent misspelling is a much more serious problem than false positives, because a spellchecker with high recall, but low precision, does not serve its intended purposes. The problem can partially be resolved by introducing grammar checking and contextual spellchecking, but this is a privilege of a few central languages only [14].

There is an old dispute about dictionary size [34, 35]. J. L. Peterson argued strongly in favor of small spellchecker dictionaries for the sake of precision. His main argument, according to his experiments, was that almost one typing error out of six may pass undetected if an English word list exceeds 100 000 elements (word-types). Quite the opposite, Damerau and Mays demonstrated with their corpus-based experiment that a 20% increase in dictionary size brings a 50-to-1 ratio between false rejections and false acceptances. Their argumentation went in favor of larger dictionaries, and that remained the dominant attitude in the spellchecking community.

We took a middle path between the arguments and attitudes expressed in the dispute. Hascheck’s dictionary is constantly growing. However, it does not grow as much as it could. Not every valid Croatian word-type form or name-type that appeared in processed texts enters the dictionary, as already explained. A good example for rejection of a valid Croatian word is the relationship between two Croatian grammatical forms. These are the adverb ‘slijedeći’ = ‘following adv.’, which is not declinable, and the adjective ‘sljedeć = ‘the following, next, subsequent, successive adj.’, which is declinable. The ErrT-file indicates that the declined forms of the adverb ‘slijedeći’, hence grammatically impossible ‘word-types’, are the most frequent errors in Croatian writing. Therefore the adverb itself was deleted from the dictionary, although it is a perfect Croatian word. According to our opinion, it is better to communicate a non-existing error, a (rare) correct use of the adverb ‘slijedeći’, than to overlook frequent errors. There are thousands of such examples, and only human language competence, supported by empirical facts, is able to detect them.

The corpus-based approach to dictionary updating is another restriction on its expansion. Only word- and name-types with confirmation in Croatian texts are in the dictionary. Theoretically, it would be possible to expand the dictionary with all allowable forms of words in it. However, we never tried to do this for theoretical and practical reasons.

Zipf’s law states that words in a corpus occur with various frequencies, where the majority have very low frequency [36]. In all languages hapax legomena (words with only one occurrence) form a dominant part of corpus types. The Croatian Language Corpus [37], a representative 85-million-token corpus of the Croatian language, contains 57% hapax legomena in its frequency profile; practically the same hapax rate (56.6%) in a corpus of similar size was reported for English too [38]. Furthermore, Zipf’s law can be extended to word forms of an inflected language. This means that various word forms of the same word occur with extreme differences in their frequencies. Some
theoretically allowable word forms have a practically zero chance of ever occurring. A proof for that is derived from a Croatian case: almost 40% of valid Croatian word- and name-types contained in a modest-size dictionary created initially for ispell—the dictionary contains approximately 200,000 different Croatian word- and name-types—are attested neither in Hascheck’s dictionary nor in the Croatian Language Corpus. This means that the probability of their appearance in Croatian is less than $3 \times 10^{-9}$ for each type, since they have not appeared in a 330-million-token corpus till now, although they all are perfectly morphologically possible. This is also a kind of language economy, and the spellchecker creator must also bear it in mind.

Practical matters are ‘hurdles’ in conventional Croatian spellcheckers. A decade ago we tested five Croatian proprietary spellcheckers with a representative sample of errors collected through Hascheck functioning. The false negative acceptance ranged between 2 and 6% [39]. The best ranked spellchecker was the Microsoft one. Recently, we tested the Croatian spellchecker for Windows 2007 with a sample of the 2000 most frequent errors common to Hascheck and the Croatian Language Corpus error profile. With this sample the Microsoft spellchecker’s acceptance of false negatives was equal to 6.25%. Besides several particular problems, the spellchecker demonstrated three systemic language problems:

1. Capitalization problem, like accepting ‘vikica’ or ‘icrc’. Correct forms are ‘Vikica’ (Croatian diminutive of female name Vicky) and ICRC (abbreviation for International Committee of the Red Cross), which are also accepted.

2. Confusion of frequent misspellings, like ‘presječe’ instead of ‘presiječe’ (present tense, singular, third person, of Croatian verb ‘presjeći’ = ‘to cut’), or ‘odječe’ instead of ‘odječ’ (inflected form of Croatian noun ‘odjeća’ = ‘clothes, clothing, dress’), with vocatives of noun ‘presjek’ = ‘cut, section’ and noun ‘odjek’ = ‘echo’, respectively. These vocatives (presječe and odječe) have zero chance of appearing in normal Croatian writing.

3. High tolerance for the so-called Serbisms in Croatian, like ‘stepen’ instead of ‘stupanj’ (both words mean ‘degree’), or ‘milion’ instead of ‘milijun’ (both words mean ‘million’). Microsoft spellchecker accepts both variants, but contemporary Croatian treats Serbisms as a profound ignorance of the language, even as an offense in writing.

With Croatian open-source spellcheckers the situation is even worse. As is usual with non-central languages, all open-source engines, ispell, aspell and hunspell, use the above-mentioned Croatian modest-size dictionary, modified on the format level only, for Croatian spell-checking purposes. All examples quoted below can be attested by using the multilingual online spellchecker Orangoo, powered by Google, aspell and TextTrust [40].

The open-source Croatian dictionary ignores:

1. Capitalization in Croatian; it allows ‘amsterdama’ to pass undetected, although the only correct spelling is ‘Amsterdama’, the Croatian genitive of Amsterdam.

2. The Croatian alphabet; it accepts ‘hydrostatičko=hydrostatic adj.’, although ‘y’ is not a letter in the Croatian alphabet, and even worse, it rejects the correct adjective spelling ‘hidrostatičko’, offering ‘hydrostatičko’ as a correction.

3. The phonetic character of the Croatian writing system; it allows ‘narezci’, presumably plural of ‘narezak = slice of ham, sausage, etc.’, but ignores that ‘z’ before ‘c’ must be converted into ‘s’; hence the correct word plural spelling is ‘naresci’.

4. Allowable Croatian endings; it allows ‘obukae’, although ‘–ae’ does not exist as an ending in Croatian; the noun ‘obuka = training’ is declined by changing the ending –a, and, in some phonetically motivated cases, by converting ‘k’ into ‘c’ before the ending –i.

Such content makes up 15% of the Croatian open-source dictionary.

Both proprietary and open-source dictionaries were obtained by deriving word- and name-types from Croatian lemmas. The process was more or less correct, and it influenced the spellcheckers’ precision. However, even perfect derivation has a systematic weakness.

Croatian nouns are declined in seven cases. The fifth case is the vocative, which is used when one calls or addresses somebody. It is semantically clear that the phrase ‘My dear Professor’ may
exist in any language, while the phrase ‘My dear microprocessor’ does not exist even in English, the best represented language on the WWW. The result is that all Croatian conventional spellcheckers leave undetected theoretically allowable vocatives, which are in most cases genuine typos, as already demonstrated above in the discussion about the Microsoft spellchecker for Croatian. The percentage of such errors is not negligible and according to our measurements all conventional Croatian spellcheckers have a false negative acceptance higher than 1% for that reason alone.

The Learning System does not use only the local context file. In many cases context broader than that locally available is needed for decision making. For that purpose the biggest text repository, the WWW, is used (Figure 1). The final result is acceptance or rejection of a word- or name-type which has been contextually tested.

At the end of the learning process, the Learning System creates two files, the word- and the name-type file for updating the dictionary. Both files are carefully checked before updating the dictionary, since the human factor in learning and supervision is also a possible source of errors. After checking them carefully, new elements are entered into the dictionary. The Learning System also updates the frequency profile of the WT-, NT-, EngT-and ErrT-files. Furthermore, it registers the text file in a corresponding log file, in terms of its system identifier (address–date–time), from which a new dictionary element was learned. No whole text is logged after passing the Real-time Subsystem for privacy reasons. The Learning System also measures the time needed for learning. Once the learning is finished, it deletes all temporary log files.

Nobody claims that Hascheck’s dictionary is 100% error-free. A special part of the Learning System takes care of errors that may occur there. If such an error is found, it is deleted from the corresponding dictionary file. The system registers each Croatian common word-type that was deleted from WT-file. Up to now 5399 such deletions were made, which amounts to 0.7% of the actual WT-file size. Each deletion in the WT-file causes updating of n-graph statistics. Since deletions in the NT-file do not cause n-graph statistics updating, they are not so precisely registered. We estimate the NT-file deletions at 1% of its actual size. This means that approximately 10 000 deletions of dictionary elements have been made up to now.

In the beginning, human supervision coped with all unrecognized tokens. With an increase in traffic it became excessively time-consuming, because of numerous error-types which were occurring repeatedly in supervised token lists. The first step for increasing learning speed was to extract from the token list prepared for supervision the elements that occur five or more times in the ErrT-file. The second step was to extract all elements that occur in the ErrT-file, regardless of their frequencies, but this step demanded a quick examination of the extracted tokens, in order to find possible wrong decisions. The third step was reduction of the supervision list of tokens that appear tagged (Figure 2), or that appear without a correction suggestion, hence at edit distance 3 or higher from dictionary elements, with the inclusion of all tokens classified in the lowest peculiarity class, as almost non-peculiar, in the supervision. The third step, which turned out to be a radical one, demanded a meticulous check-up of the list of extracted tokens, because approximately 20% of learnable name-types appeared as error-types in it. The practical aspects of these steps will be discussed in Section 4.

3. TRAFFIC DATA

Hascheck started as an e-mail embedded service in 1993, first locally, only for the staff of the Faculty of Electrical Engineering and Computing in Zagreb, but it quickly became a public service (in March 1994), primarily dedicated to the Croatian academic community. In the summer of 2003, the e-mail variant was converted into a web service with the address (http://hascheck.tel.fer.hr/).

In its e-mail phase Hascheck processed 6321 texts that formed a corpus of 39 135 406 tokens. In that phase it was a quite exclusive service with only several hundred users from Croatia and abroad, but during this phase many books and three voluminous pieces of Croatian lexicography [41–43] were processed.

With the web interface Hascheck became a broadly adopted service. Up to March 1, 2010, it processed 765 834 texts received from 78 718 IP-addresses from the Croatian IP-domain and from
Table I. Traffic distribution over IP-domains.

<table>
<thead>
<tr>
<th>ISO-3166 Code</th>
<th>IP-addr.</th>
<th>Token</th>
<th>%</th>
<th>TC</th>
<th>LI</th>
<th>ERR</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>113</td>
<td>206078</td>
<td>0.11</td>
<td>96.60</td>
<td>0.60</td>
<td>2.67</td>
</tr>
<tr>
<td>BA</td>
<td>1529</td>
<td>3040908</td>
<td>1.58</td>
<td>96.98</td>
<td>0.38</td>
<td>2.47</td>
</tr>
<tr>
<td>CA</td>
<td>32</td>
<td>499736</td>
<td>0.26</td>
<td>97.81</td>
<td>0.11</td>
<td>2.07</td>
</tr>
<tr>
<td>DE</td>
<td>201</td>
<td>617067</td>
<td>0.32</td>
<td>96.44</td>
<td>0.51</td>
<td>2.91</td>
</tr>
<tr>
<td>GB</td>
<td>65</td>
<td>277595</td>
<td>0.14</td>
<td>97.21</td>
<td>0.43</td>
<td>2.02</td>
</tr>
<tr>
<td>HR</td>
<td>73371</td>
<td>181408979</td>
<td>94.56</td>
<td>97.84</td>
<td>0.39</td>
<td>1.62</td>
</tr>
<tr>
<td>IT</td>
<td>181</td>
<td>576501</td>
<td>0.30</td>
<td>97.59</td>
<td>0.31</td>
<td>1.96</td>
</tr>
<tr>
<td>ME</td>
<td>209</td>
<td>1625887</td>
<td>0.85</td>
<td>95.89</td>
<td>0.35</td>
<td>3.70</td>
</tr>
<tr>
<td>RS</td>
<td>262</td>
<td>457209</td>
<td>0.24</td>
<td>93.86</td>
<td>0.51</td>
<td>5.16</td>
</tr>
<tr>
<td>SI</td>
<td>2064</td>
<td>1756568</td>
<td>0.92</td>
<td>94.86</td>
<td>0.66</td>
<td>4.26</td>
</tr>
<tr>
<td>US</td>
<td>126</td>
<td>701514</td>
<td>0.37</td>
<td>98.01</td>
<td>0.38</td>
<td>1.53</td>
</tr>
<tr>
<td>Others</td>
<td>565</td>
<td>671274</td>
<td>0.35</td>
<td>96.75</td>
<td>0.41</td>
<td>2.66</td>
</tr>
<tr>
<td>TOTAL</td>
<td>78718</td>
<td>191839316</td>
<td>100.00</td>
<td>97.76</td>
<td>0.39</td>
<td>1.68</td>
</tr>
</tbody>
</table>

61 other national domains. The web-phase corpus processed up to March 1, 2010 amounted to 191 839 316 tokens. All data presented in this Section refer to web-phase traffic. Some interesting data on e-mail traffic can be found in [44].

Table I shows traffic distribution over IP-domains. In Table I only countries with traffic exceeding 200 000 tokens (approximately 1 of entire web-phase corpus) are indicated by their ISO-3166 code [45].

Traffic is characterized by three basic Hascheck parameters:

- Text coverage (TC),
- Learning index (LI),
- Error rate (ERR),

which are defined as follows:

\[
TC = \left[ 1 - \left( \frac{\text{NumberOfUnrecognizedTokens}}{\text{CorpusSize}} \right) \right] \quad \text{in percentages} \quad (1)
\]

\[
LI = \left( \frac{\text{NumberOfNewTypesLearned}}{\text{CorpusSize}} \right) \cdot 100 \quad (2)
\]

\[
ERR = \left( \frac{\text{NumberOfErroneousTokens}}{\text{CorpusSize}} \right) \quad \text{in percentages} \quad (3)
\]

TC and ERR are non-dimensional parameters, while LI has a [type/token] dimension. Multiplication by 100 in (2) is introduced in order to make LI comparable to the other two parameters. The relation between three basic parameters is as follows:

\[
TC + LI + ERR \leq 100 \quad (4)
\]

In Section 4 the LI will be split into the word-type LI (LIWT) and name-type LI (LINT). It is obvious that:

\[
LI = LIWT + LINT \quad (5)
\]

TC, LI and ERR values presented in Table I are average values for the web-phase corpus.

Nearly 95% of the corpus was received from Croatian IP-addresses. The IP-address distribution is similar; Croatia takes up approximately 93% in the Hascheck address space of nearly 80 000 IP-addresses (Table I). It is worth noting that up to March 1, 2010 Hascheck had been reached from approximately 4% of all IP-addresses allocated to Croatia according to MaxMind data for February 2010 [46].
The appearance of neighboring countries, Bosnia and Herzegovina (BA), Italy (IT), Montenegro (ME), Serbia (RS) and Slovenia (SI) in Table I is something very normal and very expected. The only neighboring country which does not appear in Table I is Hungary, whose traffic is close to the threshold for inclusion in the table and amounts to 149,776 tokens, or 0.8% of the entire web-phase corpus. Relations between neighboring countries are always intensive, and the intensity is also expressed through Hascheck traffic.

Non-neighboring countries appearing in Table I are three English-speaking countries, Canada (CA), the United Kingdom (GB) and the United States (US), and two German-speaking countries, Austria (AT) and Germany (DE). Residents there are accustomed to highly functional proofreading tools for central languages. They include Croatian emigrants, children of emigrants and in Austria the Burgenland Croats, who may be trying to find something similar for Croatian too.

In all cases quoted in Table I, the error rate is much higher than the LI. This means that Hascheck sounds many more real alarms than false alarms. Such behavior explains its popularity. Table II shows traffic distribution according to the size of text received for processing.

Table II demonstrates that nearly 20% of all received texts are de facto orthographic inquiries, three-token-size on average. It is obvious that Hascheck is heavily used as an online orthographic adviser, because nothing similar dedicated to these purposes exists in Croatian web-space. A dominant part of processing is made up of texts of page size (300 tokens). E-mail size texts (45 tokens) come after them. Texts of essay (2000 tokens) or novel size (25,000 tokens) are rarely sent for spellchecking, but such texts contributed nearly 36% of the processed corpus.

Table III offers insight into user loyalty to the service. As already mentioned, only IP-addresses from which a text was sent for spellchecking are registered in Hascheck’s log files. Table III shows that no more than nine texts were sent from approximately 90% of IP-addresses. However, there are 32+9 IP-addresses from which more than a thousand texts were sent. Among these 41 IP-addresses two are Slovenian; others are Croatian, and all are corporate IP-addresses, mostly from the publishing sector. Corporate spellchecking contributes 47% to the total number of submissions, and 40% to the processed corpus. It is clear that newspaper texts are on average shorter than texts sent by ‘ordinary’ users. The ANT column gives the average number of texts processed.

Figure 3 gives traffic growth per month for the period September 2003–February 2010, or 78 months in total. The data points represent empirical data (amount of corpus that has been processed in a month), while the line is the trend-line which fits the data the best. The growth function is exponential, with the correlation coefficient $R^2 = 0.8368$, which is a high one.
However, we obtained a much higher correlation coefficient by modeling the trend of cumulative monthly traffic growth (CMT):

\[
CMT(i) = \sum_{j=1}^{i} MT(j)
\]  

(6)

where MT\((j)\) is monthly traffic for the \(j\)th month (points from Figure 3) and \(i = 1, 2, \ldots, 78\). The results are presented in Figure 4.

Figure 4 demonstrates that the CMT points for the period of the last four years (March 2006–February 2010) correlate almost perfectly with the trend-line. The cumulative approach to traffic growth modeling eliminates the seasonal oscillations which the traffic in Figure 3 has. Therefore, the traffic growth function from Figure 4 can be considered as an exact law capable of predicting...
traffic behavior in the future. It indicates that the amount of processed corpus increases on average 6.631% per month, or 2.16 times on a yearly basis.

Better arguments than those presented in Figures 3 and 4 are not needed to state that Hascheck is a very successful spellchecker. We do not know any comparable NLP system with market competition whose increase in use over a long time span has such a positive gradient as in the Hascheck case.

4. BEHAVIOR OF SPELLCHECKING PARAMETERS IN TIME

The abundance of data collected in Hascheck’s log-files allows mathematical modeling of many aspects of its life, as already indicated in the previous section. This section is dedicated to modeling the behavior of three basic spellchecking parameters: text coverage (TC), error rate (ERR) and Learning index (LI). The concept of time has to be understood relatively. We use the amount of processed corpus as the time dimension in our modeling. We also make a distinction between two ‘times’ in our modeling:

1. The web-phase corpus, which is an appropriate ‘time’ for text coverage and error rate behavior modeling;
2. The total corpus, where the web-phase corpus is shifted for the amount of corpus processed in the e-mail phase, which is the only appropriate ‘time’ for LI behavior modeling, since learning did not start with the web phase.

Empirical data will be presented again as points, but the positioning of the points has to be explained. Each point has its time-coordinate as follows:

\[ t_i = t_0 + CMT(i - 1) + \frac{MT(i)}{2} \]  

(7)

where \( t_0 = 0 \) for the web-phase corpus, while \( t_0 = 39135406 \) for the total corpus. MT and CMT are corresponding empirical values from Figures 3 and 4, respectively. Relation (7) means that each point is positioned in the middle of the month to which the point belongs. Because of differences in corpora processed per month, our months will have very different ‘durations’.

4.1. Text coverage and error rate

Figure 5 shows the behavior of the cumulative number of recognized valid tokens (CNRVT) in ‘time’. The trend-line, which correlates almost perfectly with empirical data, is a power function whose exponent is higher than 1. This means that its derivative

\[ TC(t) = 100 \cdot \frac{d}{dt} \text{CNRVT}(t) \]

(8)

has to be a constantly increasing function.

As indicated by (8), the derivative is the text coverage function. Figure 6 gives the derivative and empirical text coverage data. Because of the dense grouping of points at the beginning of the plot area, only points for the last 43 months are presented. However, the correlation coefficient in Figure 6 refers to the whole period of 78 months. We have also calculated the coefficient for the last three years, and this is higher, \( R^2 = 0.5759 \). The increase in the correlation coefficient with the increase in the corpus size proves that TC is an increasing function.

A similar procedure was applied for obtaining error rate behavior. Figure 7 gives the behavior of the cumulative number of erroneous tokens (CNET) in ‘time’. The trend-line, which correlates again almost perfectly with empirical data, is again a power function, but this time with an exponent lower than 1. Convex curves have a decreasing derivation. Since

\[ \text{ERR}(t) = 100 \cdot \frac{d}{dt} \text{CNET}(t) \]

(9)
it follows that the error rate must decrease in time. This is an unexpected conclusion, but well supported by modeling.

Figure 8 confirms the conclusion stated above. The derivation is again correlated with all 78 error rate points, although only 43 points are presented in the figure. The correlation coefficient in Figure 8 is very small, but still positive. Because of the small correlation coefficient for the whole time range, we have calculated the coefficients for two time ranges: the last three years ($R^2=0.3154$) and the last year only ($R^2=0.5426$). These coefficients are a strong statistical indication that the error rate is really deceasing.

An increase in text coverage is an expected consequence of modeling. It is a result of the word learning process described in Section 2. With an increase in dictionary size, both recall and text coverage must go up. Figure 6 indicates that the text coverage in the last few months is around
98.25%, with a trend toward approaching 98.5% in the future. In order to put these figures in the framework of conventional Croatian spellchecking, we have measured recall and text coverage of the Microsoft spellchecker for Croatian by using a representative sample composed of 8000 common Croatian words and 2000 names. The sample was supplied by a word frequency profile, which was taken from the Croatian Language Corpus [37]. The frequency profile is necessary for text coverage calculation. The results are presented in Table IV.

It is easy now, by using the content profile presented in Section 2, to calculate the expected text coverage which would be performed by the Microsoft spellchecker applied to a representative Croatian corpus; the calculated value equals 93.86%. A difference in text coverage higher than 4%-units is appropriate when one compares spellcheckers for central languages with spellcheckers...
Table IV. Parameters of Microsoft spellchecker for Croatian.

<table>
<thead>
<tr>
<th></th>
<th>Word-types</th>
<th>Name-types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall (%)</td>
<td>85.2</td>
<td>18.4</td>
</tr>
<tr>
<td>Text coverage (%)</td>
<td>99.3</td>
<td>62.4</td>
</tr>
</tbody>
</table>

for non-central ones, but not within a language. The difference also explains the popularity of Hascheck.

As already mentioned, a decrease in error rate is an unexpected consequence of modeling. Trying to explain this, we have formulated two hypotheses, one positive and the other negative. We shall start with the negative one.

It is possible that during learning we are missing a great number of false negatives, which enter into the dictionary and in that way lead to the decrease in error rate. However, we can state two strong arguments against this hypothesis, one internal and the other external:

1. Learning is a carefully supervised process and it is implausible that so many false negatives, able to harm the spellchecker’s precision so much, could pass though all controls performed by qualified people responsible for maintenance and operation of Hascheck.
2. It is also implausible that a spellchecker with low precision could have such an increase in use as is demonstrated in the case of Hascheck (Figures 3 and 4).

The second hypothesis has four components:

1. By doing spellchecking a user learns to avoid her/his systematic writing mistakes (misspellings) and in that way she/he contributes to a decrease in error rate.
2. In online spellchecking the error correcting is a much more time-consuming task than it is in conventional spellchecking. Since human behavior is based on the principle of least effort, online spellchecker users tend to be more careful when writing, in order to avoid too much correcting.
3. It is possible that many texts processed by Hascheck recently were produced by using Microsoft writing tools, including its spellchecker for Croatian, so they were already fairly clean when submitted for an additional check.
4. Experience with spellchecking can convince anyone that no spellchecker is a perfect tool. This means that a carelessly written text might contain a number of undetected errors (formally valid words, but in the wrong context as a result of mistyping) even after it has been proofread and corrected by a tool. The awareness of spellchecker imperfection also influences the care in writing, hence leads to a decrease in error rate.

We believe that the second hypothesis gives a rational explanation of behavior presented in Figure 8.

4.2. Heaps’ law and the LI

Heaps’ law is an empirical law which describes the portion of a vocabulary, in terms of different word- and name-types, which is represented in a text corpus consisting of words chosen from the vocabulary [47]:

\[ V(t) = \alpha \cdot t^\beta \]  

\( V \) is vocabulary size, \( t \) is corpus size in tokens, \( \alpha \) and \( \beta \) are parameters. With English corpora, typical \( \alpha \) is between 10 and 100, typical \( \beta \) is between 0.4 and 0.6.

Hascheck offers a unique opportunity to model WT- and NT-file growth by Heaps’ law. We did it at the very beginning of the service with a 10-million-token corpus [48], and we have been
Table V. Heaps’ law parameters for Hascheck corpus.

<table>
<thead>
<tr>
<th>Subpart</th>
<th>Heaps’ function</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$V$</td>
<td>207.24</td>
<td>0.4578</td>
<td>−778825</td>
</tr>
<tr>
<td></td>
<td>$V_{WT}$</td>
<td>662.67</td>
<td>0.3716</td>
<td>−766928</td>
</tr>
<tr>
<td></td>
<td>$V_{NT}$</td>
<td>0.4310</td>
<td>0.7467</td>
<td>123821</td>
</tr>
<tr>
<td>B</td>
<td>$V$</td>
<td>145.13</td>
<td>0.4791</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$V_{WT}$</td>
<td>4281.1</td>
<td>0.2722</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$V_{NT}$</td>
<td>0.006014</td>
<td>0.9701</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>$V$</td>
<td>819.9</td>
<td>0.3852</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$V_{WT}$</td>
<td>5398.1</td>
<td>0.2592</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$V_{NT}$</td>
<td>6.8616</td>
<td>0.5885</td>
<td></td>
</tr>
</tbody>
</table>

doing it regularly every several months in the web phase. Besides WT- and NT-file growth ($V_{WT}$ and $V_{NT}$ functions), we also model the whole dictionary growth ($V$ function), allowing

$$V_{WT}(t) + V_{NT}(t) \neq V(t)$$

for reasons of best fit. This means that we independently model the increase in size of two variable dictionary files and the total increase in these two files. The results are presented in Table V.

Parameters in Table V are given for three total corpus subparts:

(A) Initial 10-million-token corpus in the e-mail phase;
(B) Corpus between 50 and 100 million tokens;
(C) Corpus over 100 million tokens.

As Hascheck did not start operating with an empty dictionary, we had to introduce a shift parameter $K$ in Heaps’ law for the 10-million-token corpus:

$$V(t) = \alpha \cdot (t - K)^\beta$$

which expresses the size of the corpus used for obtaining the initial dictionary of approximately 100,000 Croatian common word-types. Approximately 80% of the one-million-token ECLC is its Croatian part. By applying (12) it is easy to obtain $V(t=0) = 103,177$ [type] and $V_{WT}(t=0) = 101,875$ [type]. Both results confirm the initial dictionary size. The positive value of parameter $K$ in the case of $V_{NT}(t)$ means only that in $t=0$ the NT-file was empty. It also shows that name-type learning started shortly after introducing the service into its operational phase. For web-phase corpus modeling no $K$ shift was needed anymore. Since Heaps’ law is a cumulative law, the correlation between empirical data and functions from Table V was always over 0.99, similar to the three cumulative law functions already applied in this paper (Figures 4, 5 and 7).

At the whole vocabulary level ($V$ function), parameters $\alpha$ and $\beta$ for subparts A and B do not differ very much. Parameter $\alpha$ is several times higher than in the English case—the difference expresses the inflectedness of the Croatian language compared with English—and parameter $\beta$ is between 0.4 and 0.6, as it is in English. A proof that there is no great difference in $V$ functions is seen from the following calculation:

- by applying the parameters for the 10-million-token corpus at $t = 10^8$ tokens we obtained 955,926 [type];
- by applying the parameters for the 100-million-token corpus at $t = 10^8$ tokens we obtained 987,546 [type].

Taking into account that the second calculation is the correct one, the first calculation has an error of $-3.2\%$.

However, subpart A and B parameters differ significantly in both the $V_{WT}$ and $V_{NT}$ function. With the increase in corpus size, the parameter $\alpha$ for the $V_{WT}$ function grows, whereas parameter $\beta$
drops. In the case of the $V_{NT}$ function the parameters demonstrate the opposite behavior. Parameter $\beta$ is much more influential on function growth than parameter $\alpha$. In the case of the function $V_{NT}$ parameter $\beta$ tends toward 1, and $\beta = 1$ means linear growth.

Kornai has exhibited a mathematical proof that the vocabulary of any natural language is potentially infinite [38]. Here we have an explanation for that. Names, most of them imported from other languages, make vocabularies potentially infinite, while the common word vocabulary of a language tends, because of $\beta_{WT}$ decrease, toward a kind of saturation with corpus size growth, as demonstrated by the data from Table V.

While differences in parameters for corpus subparts A and B are influenced by corpus growth only, a change in parameters in subpart C is strongly influenced by the radical change in learning strategy described at the end of Section 2. This is well expressed by changes in LI behavior presented in Figures 9, 10 and 11, respectively.
The LI, defined by (2), can be expressed according to Heaps' law as follows:

\[ \text{LI}(t) = 100 \cdot \frac{dV}{dt} \quad (13) \]

It is clear that \( \text{LI}_{WT} \) and \( \text{LI}_{NT} \), the LIs for word- and name-types (Figures 10 and 11), can be obtained in the same manner.

It is obvious from Figure 11 that the radical change in learning strategy (step 3 in Section 2) caused a discontinuity in learning name-types. Constant \( \text{LI}_{NT} \), characteristic for subpart B in Figure 11, became untenable with the increase in traffic described in Section 3 due to workload reasons. After the experience with steps 1 and 2, when they do not contribute sufficiently to a decrease in \( \text{LI}_{NT} \), we decided to introduce a radical change, sacrificing learning most name-types that occur only once. We obviously succeeded in that, since parameters for the \( V_{NT} \) function in Table V (subpart C) have ‘normal’ Heaps’ law values. It is worth noting that sacrificing the learning of single-occurrence names did not harm text coverage and error rate (Figures 6 and 8). It is also worth noting that the radical change in learning strategy did not cause any discontinuity in common word-type learning (Figure 10), which was our intention too. The discontinuity in the LI function (Figure 9) is a result of discontinuity in the \( \text{LI}_{NT} \) function only.

Figures 9–11 give empirical data for the last three years (March 2007–February 2010) of processing. The LIs of that period are interesting for the elaboration that follows in Section 4.3.

Correlation coefficients presented in Figures 9, 10 and 11 are much higher than those presented in Figures 6 and 8. This is so because Heaps’ law is much less influenced by human behavior than are the cumulative data functions (Figures 5 and 7) from which the text coverage and error rate function have been derived. The lower correlation coefficient value obtained for the \( \text{LI}_{NT} \) function confirms only that name-type behavior in a corpus is less predictable than common word-type behavior. A high correlation coefficient \( (R^2 = 0.9301) \) obtained for the LI function makes this applicable for workload anticipation.

4.3. Anticipation of learning volume

The possibility of anticipating volume of learning is crucial for maintenance of a supervised learning system. The anticipation can tell the maintainer how many man-hours per certain period of time are needed in order to perform the complete task of learning. This is extremely important in cases of non-funded projects as many spellcheckers for non-central and under-resourced languages have turned out to be. The particular reasons for not funding Hascheck in its developmental phase...
are given in [44]. Since workload in the operational phase has never exceeded 40 h monthly per maintainer, we have never asked for any type of funding later. However, the Agrokor Group, the largest private company in Croatia and one of the leading regional companies, recognized Hascheck as an infrastructural service valuable for Croatia and gave it a $40 000 grant in March 2010.

Learning volume (LV) equals:

\[ \text{LV} = \frac{(\text{LI} \cdot \Delta t)}{100} \] (14)

where LI is the learning index and \(\Delta t\) is the corpus volume that should be processed in a certain time interval of the future. Both values are well predictable for reasons presented in Sections 3 and 4.2, respectively.

Based on parameters from Figure 4 and taking into account seasonal traffic oscillations from Figure 3, we have developed a simple but very reliable algorithm for estimating monthly traffic for the coming months. The estimated monthly traffic (EMT) is calculated as follows:

1. If \((\text{month} = 9)\) \{\(\text{EMT(\text{month})} = 2.16\times\text{MT(month-12)}\)\};
2. For \((\text{months} = 10, 11, 12, 1, 2, 3, 4 \text{ or } 5)\) \{\(\text{EMT(month)} = 1.1\times\text{MT(month-1)}\)\};
3. For \((\text{months} = 6, 7 \text{ or } 8)\) \{\(\text{EMT(month)} = 0.9\times\text{MT(month-1)}\)\}.

The algorithm states the following:

1. Estimated monthly traffic for September is equal to monthly traffic (MT) exhibited in September a year ago multiplied by 2.16.
2. For the period from October through May of next year estimated monthly traffic goes up by 10% each month.
3. For the period from June through August estimated monthly traffic goes down by 10% each month.

In the algorithm we have used empirical MT data from Figure 3 and by applying it to the whole range where EMT is defined we obtained an estimation error of only 0.5% on the cumulative level.

After obtaining EMT it is easy to calculate the corresponding LI value. With the function from Figure 9 in hand it is only a matter of appropriate time-coordinate positioning:

\[ t = 39135406 + \text{CMT}_{\text{before}} + \frac{\text{EMT}}{2} \] (15)

where \(\text{CMT}_{\text{before}}\) refers to the total web-phase corpus processed before the month for which the estimate is made. Hence, the calculated LI, together with \(\Delta t = \text{EMT}\), is used in formula (14), in order to obtain the learning volumes that should be expected in the coming months.

Figure 12 presents the results of the anticipation procedure. The results are given for the last three years, since starting with March 2007 the monthly traffic went over one million tokens per month, thus causing a substantial increase in workload by learning. Gray bars are real LV values, while black bars are expected LV values.

This figure demonstrates that there were more overestimations than underestimations of learning volumes, which is a positive fact from the aspect of workload planning. Only three overestimations (and not one underestimation) exceeded 25%, with regard to the real volume of learning performed in the corresponding month. This happened in November 2007 (bar 9), February 2008 (bar 12) and February 2010 (bar 36). Perhaps we should refine our EMT algorithm by taking into account that February is the shortest month of the year. However, even without that, the correlation between real and anticipated values \((R^2 = 0.8985)\) confirms the applicability of our anticipation procedure for purposes of workload planning.

Another important dimension of workload planning is learning speed (Figure 13).

With learning volume ranging under or around 10 000 types per month we could sustain the learning speed of approximately 200 types per hour. Our anticipation of learning volume increase motivated increases in learning speed, which were achieved by changes in learning strategy described at the end of Section 2. Step 1 was introduced in April 2007, and it increased the
learning speed to approximately 250 types per hour. However, this increase was demonstrated to be insufficient very soon. In May 2007 (bar 15 in Figures 12 and 13) we had the highest workload of all; it approached 90 h spent on supervised learning in that month. Therefore Step 2 was introduced in October 2008 (during the summer of 2008 there was no fear of learning volume increase for seasonal reasons). Step 2 increased the learning speed to approximately 300 types per hour, but it too was demonstrated to be insufficient. By introducing the radical Step 3 in April 2009 we reduced name-type learning drastically (Figure 11) and increased the learning speed to approximately 350 types per hour (bar 26 at Figure 13). A further increase in learning speed to over 400 types per hour, exhibited in January and February 2010 (last two bars in Figure 13), is not a result of any change in learning strategy but a consequence of improvements in software that support learning [49].
With the end of the year 2010 the Hascheck corpus will reach 400,000,000 tokens. From January 2011 onwards we expect traffic of over 25,000,000 tokens per month. In this range of corpus and traffic the learning volumes will exceed 40,000 word- and name-types per month, meaning that the workload should go over 100 h per month if nothing changes. Therefore, we plan to introduce registration of Hascheck’s users by autumn 2010. The registration will allow a user to create and maintain a user-defined customized dictionary as an extension of Hascheck’s dictionary. With the introduction of user-defined customized dictionaries the workload caused by supervision of learning will disappear. This saving will be used for accelerating the development towards contextual Hascheck, which is affordable for a spellchecker that operates with a dictionary derived from a corpus amounting to nearly half a billion tokens.

5. THE FUTURE

We shall not discuss Hascheck’s future here, which is more or less predictable, but the future of online spellchecking in the context of under-resourced and non-central languages, the huge majority of languages of the World. Part of our consideration will be extended to central languages as well.

In the case of under-resourced languages, the development of a spellchecker is a question of survival. An under-resourced language usually exists in conjunction with a central language (Chinese, French, English, Russian, Spanish, etc.), which puts pressure on official written communication. Very often such a language does not have a standardized writing system, and spellcheckers bring in the first de facto standards. Scannell has described how to obtain an initial spellchecker dictionary for under-resourced languages [15]. However, if the dictionary is used inside a conventional spellchecker only, it remains static and has low linguistic functionality. Hascheck-like use takes advantage of the spellchecker users for dictionary upgrading. The result is returned to the user community in the form of increased linguistic functionality. As described in this paper, a 10-million-token corpus has to be processed in order to increase text coverage from approximately 85 to 95%. Such a corpus is not an enormous corpus. To remind ourselves, Hascheck started operating as the first spellchecker for Croatian, hence as an under-resourced language spellchecker, with text coverage of 87% on average for the first three months of public functioning. The text coverage increased relatively quickly to 95% [44].

In the case of non-central languages, research and development will turn mostly to contextual spellchecking. This paper has described extensively how context can be used to upgrade non-contextual spellchecker functionality without losing its precision. However, this is possible, with a workload acceptable for the researchers, only within online spellchecking. There is only one step from the use of context for human supervision of learning to context use in automatic spellchecker functioning. The step is not simple, but it is mostly technical. Furthermore, contextual spellchecking will cope not only with real-word error detection and correction problems, but also with algorithms that improve suggestion adequacy for non-word corrections based on the context in which non-words appear, as is well demonstrated in [50]. For both purposes the rich log files obtainable through non-contextual online spellchecking are of great value. Finally, in the framework of context-dependent spellchecking, much more attention will be paid to name spelling, because the spelling of personal and other names, which is much more variable than that of common words, can be successfully verified and, if necessary, corrected by examining the context. New compilations of spellchecker dictionaries will also tend to have more and more new name-types, because this is the only way in which the overall recall of a mature spellchecker can be improved. Hascheck is a good example of this.

Online spellchecking might become the dominant form of the service in the future. This speculation is based on the characteristics of living languages and on the demands arising from human writing. Human writing, as an individual and creative act, is very variable. It may be multilingual (scientific writing, usually involving the use of a technical sublanguage); it may require the extensive use of proper names of various origins (journalism), or intensive use of archaisms, slang,
dialect, obscenities, spelling variants (fictional writing) and so on. None of the existing conventional spellcheckers can satisfy all these requirements at the same time. On the other hand, a language deserves to be called alive if and only if it constantly produces new words as a result of changes in its surrounding reality. At the same time some words are passing away, becoming a part of the archaic instead of common human vocabulary. All the changes and variations have to be registered instantly, in a linguistically proven way, in computerized dictionaries of living languages. This is possible only with systems which monitor what is happening inside a language on a daily basis, hence only with online spellchecking.

Coordinated online spellchecking might become a kind of a parallel language banking system needed in order to reach the ultimate NLP goal, global speech-to-speech translation [51].

Let us imagine a situation in Europe. Europe is taken as an example because the European Union’s concern about multilingualism has materialized in rich funding for NLP projects that aim to approach, as closely as possible, to the ultimate goal. The funding has resulted in many brilliant breakthroughs, but the goal is still far away. The main task of an imaginable European Language Banking System (ELBAS) should be an acceleration of the existing research towards the goal. This could be done if research projects from various NLP fields (Automatic Speech Recognition, Machine Translation, Speech Synthesis and others) and countries are supplied with parallel and accurate data. Hence, the first duty of ELBAS should be to catch languages on the fly, in their everyday changing use, in a methodologically and technically unified way. On the level of written language, it could be done by a network of Hascheck-like systems, as presented in Figure 14.

National systems should be free-of-charge, but with a right authorized by their users to collect their texts for research purposes. Our experience with Hascheck’s learning has shown us that texts dealing with the same subject appear in various languages every day. They are easily recognizable by the strange names appearing in them. National systems, in Figure 14 presented as Hascheck (Croatian), Pascheck (Portuguese) and Plascheck (Polish), should supply EUscheck, the system under direct control of ELBAS, with a daily collection of texts marked by a text subject identifier and corrected, if necessary. In EUscheck they should be classified into the so-called quasi-parallel text classes using criteria of subject similarity, which often appear as a number of identical names in the texts. We believe that such a kind of daily accurate data acquisition could significantly
accelerate MT in Europe. The idea can be extended to the whole World too, under the auspices of the United Nations.

The latter ideas are probably only our dream, but online spellchecking, even without that, has its perspectives. Instead of spellchecking engine competition, which we witness today on the application level of spellchecking, the future might bring competition—but also cooperation—among various dictionaries distributed over the network as a kind of knowledge-based resource, maintained by linguistically qualified specialists, which the user’s writing tool can invoke automatically, as many as needed, depending on the characteristics of one’s writing. Finally, the public character of language writing systems figures as a strong argument in favor of the future of online spellchecking. Anything involving identity issues must not be treated as a proprietary matter but as a public service, to be paid for, if necessary, not by licenses but on a pay-per-use basis.

6. CONCLUSION

The future of spellchecking might be online spellchecking. There are several reasons for that:

1. The public character of language writing systems and identity issues connected with language use, which argues against a proprietary character for spellchecking tools.
2. The need to overcome existing gaps in basic NLP tools between central and non-central languages, and even under-resourced languages.
3. The shortcomings of open-source spellchecker dictionaries, which are static and often encumbered with low linguistic functionality.
4. The necessity for including spellchecking in the main NLP task of the world, to get speech-to-speech translation functioning on a global level, not only for a few privileged world languages.

This paper has described in detail what online spellchecking, as a collective and shared endeavor, can do for an under-resourced and non-central language which is still in the process of standardization. Hascheck is today the de facto standard in Croatian spellchecking, proven by its intensive use by journalists. They produce 40% of its traffic, which is measured in millions of tokens per month. After reaching linguistic functionality comparable to that of conventional central language spellcheckers, Hascheck is one step away from contextual spellchecking. This would have been impossible in Croatian without the online approach to spellchecking. Finally, the workload data presented in this paper prove that online spellchecking can be done with zero extra funding, as a secondary job along with everyday duties. We consider this an encouraging fact for researchers and developers who aim to create the first spellchecker for an under-resourced language, or to improve spellchecker functionality in non-central languages.

REFERENCES

2. GNU Aspell. Available at: http://www.aspell.net/ [25 June 2009].
