Network Problem Diagnostics using Typographic Error Correction

Martin Holkovič
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
iholkovic@fit.vutbr.cz

Michal Bohuš
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

Ondřej Ryšavý
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
rysavy@vutbr.cz

***Abstract*—Detecting and correcting network and service availability issues is an essential part of the network administrator's daily duty. One of the causes of errors can be the user herself providing incorrect input. The present work describes a new diagnostic method that detects incorrectly inserted inputs observed in network-related data, e.g., network traffic, log files. The proposed method aims to detect incorrect words in domains, login names, or email addresses. First, we describe how to detect possible incorrect words. For each such detected word, a list of correct candidates is created based on edit distance. Next, the correction method selects the best word by scoring candidates based on the probability of occurrence in the given context. The proposed method was implemented as a prototype and tested on words created using real user activities. The evaluation demonstrates that this approach can substantially reduce the time needed to identify this kind of errors.**

***Index Terms*—computer network errors, network diagnostics, typographic error correction, end-user data diagnostics**

I. Introduction

Errors in computer networks can be caused by a lot of different types of faults. System or device failure can prevent networks from working, some services may become unavailable, and user experience could be negatively affected. Due to the great variety of errors, there is no single procedure or path to detect the cause of all network-related errors.

Some of the errors can be caused by incorrect user input. For example, users can often type a wrong URL into a web browser, wrong credentials into a mail transfer agent, or an incorrect phone ID for a VoIP call. When a user incorrectly specifies some of this data, the requested service will be unavailable or not work as expected. Because some of the information entered by a user is necessary to initiate a network connection or is transmitted in network messages, it is possible to detect wrong information by analyzing of suitable data sources, e.g., network packets, NetFlow records, or log files.

We propose a new diagnostic method that applies typographical error detection techniques and correction commonly used in spell-checkers. The proposed method can inform the network administrator about the incorrectly inserted values from end-users by detecting typographical errors in user data. At the same time, it provides a suggestion for a possible correction. This can be useful for the administrator as he can automatically find out whether the error is created by a typographical error or if there is another reason.

The paper contribution is as follows. We developed a method for detecting errors in user data. The method is similar to that used by spell-checker systems. For each possible incorrect word, the algorithm creates a list of possible correct word candidates selected using the number of edit operations as a distance between the words. Among these candidates, the correct word is selected using the ranking method. The problem with current solutions is that they do not work well without context (individual words) and with words that are not based on grammar rules (e.g., email address xholko00@fit.vutbr.cz).

The method's direct application is to validate typographical errors in domain names and usernames, which enables us to apply it to a wide range of applications and services. Regarding domain names, except for the purely diagnostic use case, the method can be employed for detecting various types of malicious activities, such as an IDN homograph attack, typosquatting, and other forms of domain phishing attacks.

This paper presents the principles and design of a new diagnostic tool that focuses on detecting typing errors. The method relies only on passive data sources (captured traffic, NetFlow records, log files). Similarly, as in [8], our method will be able to work in a learning mode for which it will require error-free data. The proposed method is considered as a complementary tool for the existing systems.

The paper is organized as follows: The next section describes related work mentioning the key results in computer network diagnostics. Sec. III provides background information about research done on spell checking. Sec. IV defines the principles behind the developed tool. Sec. V presents the architecture of the tool. In Sec. VI, the approach and results of the evaluation are provided. Finally, Sec. VII concludes the paper by discussing the contribution and identifying further improvements.

II. Related Work

Because computer networks are complex systems, errors are unavoidable and sooner or later occur [1]. Errors can affect network performance or user experience, which can cause other network problems [2]. Therefore, it is necessary to find and correct errors correctly. This is addressed in diagnostics, which is an important part of network management [1].

Fault diagnosis is a time-consuming activity that requires in-depth knowledge of network operation. Administrators often do not have the appropriate tools or knowledge to diagnose network problems, and they would like to have sophisticated automated tools to help them diagnose those errors [3]. In case of insufficient automatic tools, problems must be diagnosed manually, for example, using the Wireshark tool [4], [5].

Network problems can be divided into application and network problems [6]. An example of application problems is a broken server service or a badly configured client software. Problems related to network infrastructure fall under network problems. An error can be caused by a human - unintentional (misconfiguration) / intentional (attack), or a device failure [7].

Currently, a single tool that can diagnose all kinds of errors does not exist. Researchers have so far developed a wide range of diagnostic tools [8]. The tools can be divided either according to what data they work with - network packets, SDN data, NetFlow records, log files [9]–[12], or how they access the data - passive, active, and hybrid [6], [13], [14].

III. Background

Researchers are working on correcting typographical errors since the 1960s [15]. Correcting this type of error is based on the fact that people make mistakes when typing input data. Because users are often unaware of the error, a wrong value can cause a problem. One of the best-known examples of typographic error correction in misspelled words is in text editors such as Microsoft Office Word [16]. There are also other uses, such as Optical Character Recognition (OCR) [17] or typographical error password tolerance [18], [19].

Kukich [15] has divided the problem of typing errors correction into three categories:

1) nonword error detection (wrong word detection only),
2) isolated-word error correction (finding the wrong word and proposing a correction),
3) context-dependent correction (finding the wrong word and offering a correction based on the text's context).

Word correction extends error detection by offering correction for the word with a typographical error. It tries to guess what word the user could have thought of, solves candidates' selection, and possibly chooses the best candidate. In addition to common language words, searching for typographical errors also makes sense in other data types such as numeric values [21] or domain names [22].

Correcting typographical errors consists of three parts: 1) finding an error; 2) creating a list of candidates; 3) rating of individual candidates [15].

A. Finding an error

To detect a wrong word, it is necessary to model the words of the language. The model needs to detect several sources of error, such as pronunciation similarity, typographic similarity, or user's bad habits [23]. The most commonly used models are based on vocabularies or n-grams [15].

1) N-grams: N-grams are n character long substrings of words. The most commonly used are bigrams (n=2) and trigrams (n=3). Finding errors based on n-grams works by generating all n-grams from the analyzed word and comparing them with a previously created model. If the n-gram is not present in the model or only with a small occurrence, it is expected that this is an error. An example is a "zfq" trigram, which is not common in English. To create an n-gram model, a huge text containing only words without errors is needed.

2) Dictionaries: In this case, the model is the dictionary itself, created by storing all the unique words from the error-free text. If a word cannot be found inside a dictionary during the check phase, it is considered a typing error [24], [25].

A search response time may be a problem when using the dictionary as a model [18]. A common technique to speed up dictionary searches is to split one large dictionary into several smaller ones. One way to divide is by word length [31]. For example, when searching for a word of 5 characters in length, it will only search within words of the same length.

The basic and often used technique to access a dictionary is by using a hash table. A dictionary can also be implemented by tree structures such as a ternary search tree [32], trie, binary search trees sorted by frequency, trees with words or characters in nodes, bloom filters [33], or finite state machines [34].

3) Types of errors: The two basic types of errors that automatic word correction focuses on are [32], [35]:

- nonword error - if an error occurs, a non-existent word is created (e.g., hello - henlo);
- real-word error - an error will result in an existing word (e.g., pay - day).

Errors that result in an existing word are more challenging to detect because it is necessary to know the text's context. Mitton analyzed the errors from tests filled out by students at the age of 15 and found that the real-world error accounted for 40% of the errors [36]. The types of errors vary depending on the environment in which they occurred; for example, if it is a writer, most of the errors will be due to pressing the wrong key. On the other hand, OCR errors will be based on similar-looking characters such as B and 8. Word-bounded errors are a specific kind of error where the space between words is missing, or space is shifted [32].

There are three kinds of mistakes that result in a typographical error [15]:

- typographical errors are those where the user types the word "henlo" instead of "hello", where it is assumed that the author knows how the word is spelled and only pressed a bad character on the keyboard;
- cognitive errors arise from insufficient knowledge of the user - the user did not know the correct form of the word;
- phonetic errors occur when replacing a character with another similar-sounding one.

B. Creating a list of candidates

When a misspelled word is detected, it is determined from which possible words the misspelled word could have been formed. For this reason, functions that determine the distance

between two words are being used. The earliest and one of the best-known ways to determine the distance of two strings is the Levenshtein distance [37]. The principle is that we specify the minimum number of operations required to transform one string into another. Allowed operations are insertion, deletion, and substitution. Wagner and Fisher [38] also called the distance the editing distance. There are other types of editing distances, such as Hamming [39], [40], Dameraou-Levenstein [41], or Spring-Winkler [42].

The test with data from Birkbeck Spelling Error Corpus [43] showed that out of 1000 errors in the English language, 742 are one-character, 201 two-character, 44 three-character, 9 four-character, 4 five-character [24]. In the case of TshwaneDJe Sesotho sa Leboa corpus [44], from the 908 errors in domain names were 804 one-character, 78 two-character, and 26 three-character. Further analysis of large texts' errors confirmed that 80-95% of errors are single-character errors [27]. The results show that it is most important to check words with a short editing distance when searching for errors.

C. Rating of individual candidates

After generating all words from which it is possible to create a searched word with a typographical error within a certain editing distance, it is necessary to rate them. The purpose of the rating is to find the most likely word from which the error originated. Research is focused not only on the accuracy of results but also on the speed of the ranking process [45], [46].

If the user interactively selects one of several options, it is possible to adapt to a specific user and his type of errors [47]. Another learning option introduces a learning mode during which all the found words are saved as correct [25]. In addition to classic methods, it is also possible to use machine learning and artificial intelligence [48], [49].

Even though it is impossible to use the language's grammar to find the best candidates for correction [31], [50], it is still possible to use the probabilities of specific errors to prefer some words before others. The most basic heuristic is to adjust the weights of the editing distance according to the type of operation. For example, several sources suggest that each operation has a different probability from others. Specifically, their probability is in the following order (from most to least probable): substitution, deletion, insertion, and transposition [36], [44], [51]–[53].

IV. Methodology

This work aims to create a new method for searching for typographical errors and proposing their correction in data that contain settings and parameters of various network applications. It does not make sense to analyze all data that are coming from the user. For example, hashed passwords, timestamps, or any random values. In contrast to detecting typographical errors in plain text, the method must work with words for which it is impossible to apply grammar from plain text, such as domain names or IP addresses. Simultaneously, all words will be processed individually, so it will not be possible to use the context in which the words are located. Since the typing error is not created by a machine but by a human, it is necessary to focus on the user's data. We have recognized the following categories of data: IP address, transport port, domain name, username, generic number, generic string.

The term word in this work means a character string belonging to one of the described categories. It will also be possible to create combined categories from these categories. An example of a combined category is an email address consisting of a login name, a "@" delimiter, and a domain name. With combined categories, the values are broken down into basic categories and stored in the appropriate dictionaries. This approach allows knowledge sharing, so if someone on the network visits the "company.local" domain name, the system can find a typographical error in the "user@copmany.local" email, even if the email has never been seen before.

A. Detecting an error

Our proposed method works on the principle of dictionaries. What is in the dictionaries is valid and correct; what is not is considered a typing error. N-gram analysis could also be used for this purpose, but for words that are not part of the natural language (domain name, IP address), the N-grams would not work correctly. Many correct words would be marked as incorrect, and at the same time, many incorrectly marked as correct. Dictionaries are also used to filter out possible corrections for detected incorrect words, and a suitable replacement is eventually selected using heuristic methods.

Words are not all stored together but are stored by word category in several smaller dictionaries. This means that only a dictionary containing email addresses is used to determine if an email is a correct word or a typing error. Each dictionary further consists of two parts - predefined and learned.

The predefined part of a dictionary contains words that are expected in a given context. In the case of words of a common language, this is a list of all grammatically correct words. This can be a list of the most visited domain names or transport ports of well-known services for other data types. These dictionaries are expected to be customized by the administrator based on the data that are present inside the network. For example, the administrator can enter all company's usernames.

The dictionaries' learned parts are not created manually by administrators, but their content is created by learning from the input data. One way to teach a tool the correct data is to use only data for learning that is not reported as problematic. The downside is that end-users do not report all problems because they can fix some typographical errors themselves. The second way of learning is that the administrator manually verifies which data is correct and which is not. However, this is unrealistic due to the huge amount of network data in real networks. The last option is to use an external tool that evaluates each communication, whether it has finished successfully or with an error [55]. In this case, the tool would automatically learn only from correct communications.

B. Generating candidates

For all words detected as words with an error, finding suitable candidates for correction in the dictionaries is neces-

sary. Candidates are all possible words from which the typing error could be created with a certain editing distance. We are working only with words that have an edit distance of less than or equal to 2. Simultaneously, to consider a word a correct candidate, the word needs to be in the appropriate dictionary.

The creation of candidates is based on applying reverse operations to basic operations (insert, delete, replace, and swapping of two adjacent characters). For example, if we want to determine if the word "ello" was formed by omitting the first letter, the method tries a reverse operation (in this case, the opposite of omitting is the insertion of a character). After trying all characters, the method detects that inserting the letter "h" before the word "ello" creates the valid word "hello". To try all the possibilities, it is necessary to try all operations on all word positions and to use all combinations of characters. Candidates with edit distance 1 are created by one operation and candidates with distance 2 by two operations.

Our method uses two different ways of creating candidates. The first method applies all operations to the detected word with an error and tries to obtain words present in the dictionaries [3]. The second method works oppositely and takes correct words from dictionaries, and calculates the number of operations required to create a correct word from the misspelled word. The second method is more complex, but with fewer words in the dictionaries, this method is significantly faster. The amount of words for which the second method is faster than the first one is described in the evaluation section.

C. Choosing the best candidate

After creating candidates for word correction, it is necessary to determine the best replacement from them. The easiest way would be to select the candidate with the shortest editing distance. The problem is that several candidates may have the same minimum distance. Based on this information alone, we cannot select the most suitable candidate.

We propose a scoring algorithm that tries to assign higher scores to the candidates with higher probability. At first, the algorithm assigns a basic score for each candidate. In the next step, the algorithm takes the operations that are needed to change the word with an error to the correct word (e.g., insert letter "h" at the beginning of the word and replace the third letter "i" with the letter "l"). For each of the operations, a penalty is deducted from the score. The main idea is that each operation has a different penalty based on probability.

We have analyzed several research papers and identified which errors are more probable than others. The list of these errors is in Table I. If the operation contains any of these errors, its penalty is reduced. Each operation may be attributed to several probable errors, and the penalty is reduced for each of them. For example, an error in the username "matousek", where the user mistakenly writes "mat0usek" is represented as an operation to replace the fourth character "o" with "0". This operation contains up to three errors with a higher probability: 1) error is not in the first character; 2) characters "o" and "0" are in a fat finger distance; 3) characters "o" and "0" are visually very similar. All errors have the same weight, and the same amount reduces the penalty for each of them.

TABLE I
LIST OF ERRORS WITH A HIGHER PROBABILITY

Error type	Example
Duplication [21]	44 > 444
Shift [21]	441 > 411
Pluralism [15]	money > moneys
Doubling or skipping a voice [36]	scissors > sissors
Fat-finger [56], [57]	help > hewlp
Transcription error [19]	1 > l
Phonetic error [32]	blake > brake
Keyboard layout [19], [32]	zoomba > yoomba
Capitalization [19]	home > HOME
Error not in first character [31]	config > comfig

V. TOOL ARCHITECTURE

This section describes the architecture of the designed tool, which implements the proposed diagnostic method. The tool is intended to help network administrators with finding and fixing a problem on the network. A typical use case for this is when a service does not work for an end-user. The end-user reports the problem to the administrator, who uses data analysis to determine that the service is not working for the user because he entered some data incorrectly due to a typing error. Finally, incorrect words with possible corrections will be displayed to the administrator. Before using the tool, the administrator needs to know what data to analyze and in which location this data can be found.

The proposed tool consists of several independent blocks, shown with their interconnection in Fig. 1. In the left part of the figure, there is the input data, which, after processing, continues to other blocks according to the selected mode. The result of the first (learning) mode is a set of learned words, and the result of the second (diagnostic) mode is a list of detected words with a typographical error (or errors) and the proposed corrections.

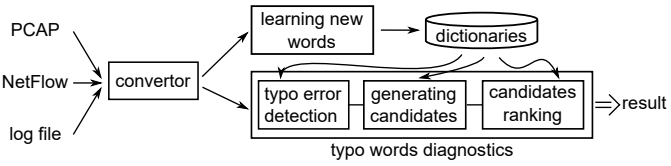

Fig. 1. An architecture of the proposed tool which consists of multiple blocks.

A. Convertor

The first part of the tool is used to convert data from different sources that use a different format into a common format. The common format will allow the rest of the tool to work with data no matter its source. For this purpose, a converter is used. In addition to data converting, it filters the input data because it does not make sense to analyze all the data from the sources (e.g., timestamps or password hashes). The format is based on JSON, consisting of keys (categories)

and lists of values belonging to those categories. The rest of this subsection describes currently supported sources.

1) Network packets: The analysis will look at user data entered into the application and transported in network packets. An example is the configuration of an email client, where the user enters the login name. Packets can be saved in the PCAP format for later analysis. We are using TShark to convert packets saved in PCAP format into JSON format, which is better for data analysis. We have chosen TShark because we do not need to implement our custom protocol parsers.

2) NetFlow records: Another way to analyze network data is to use NetFlow technology, which aggregates packets according to common properties and reduces the amount of analyzed data. The tool is working with NetFlow records that are saved in the CSV format. This CSV format consists of columns representing NetFlow attributes and rows that contain individual records. It is important to note that current state-of-the-art implementations of NetFlow monitoring are not sampling the analyzed data even for high-speed networks and are exporting application fields, e.g., the domain name from DNS or email address from SMTP.

3) Log files: The advantage of processing log files over network data is that packet analysis cannot process any data in the case of using an encrypted connection. However, the data is written to the log files in a readable form. Processing log files allows the analysis to look for errors regardless of the network topology or protocols used. Although log files are in text form by default, their format is not uniform. Each application creates logs differently, and therefore it is up to the administrator to specify exactly how the information should be searched for and extracted from the data. For this purpose, we used regular expressions to specify which part of the record will be analyzed.

B. Learning mode

After the converter processes the data, it is possible to start one of the two modes. The first mode is responsible for learning new words and saving them into dictionaries. Words are stored in the appropriate dictionary according to their category and length. For example, when saving the domain name ”milk.com”, the word’s length is determined (it is 8) and saved to the dictionary containing only domain names of the same length. The goal of dividing the dictionaries by word length is to make it easier to find words with similar lengths.

C. Diagnostic mode

After the tool has learned the correct words, it is possible to proceed to the diagnostics mode. The second possibility is that the learning mode was not used at all, and the administrator manually predefined all the correct words. This consists of three parts - typing error detection, generating candidates, and candidate ranking. Each of these parts is using dictionaries that contain both predefined and learned words.

1) Typing error detection: The first step in diagnostics is to find out which words are correct and which words have a typographical error. To do this, it is necessary to load dictionaries of correct words. When the program is loading dictionaries, the individual words are loaded into Python sets. The search for whether a word is in the dictionary is performed as a search for an element in a set. The benefit of implementing a search over a set is that it is possible to search for multiple elements in a very short amount of time. In our test, we searched for 1000 elements in less than 1 ms, while the data structure with 100.000 words took up less than 16 MB of operational memory.

2) Generating candidates: The second step in diagnosis mode is to obtain potential candidates to correct the detected typographical error. The proposed method is using two algorithms for generating candidates. The first one is applying operations to the word with a typographical error to create a correct word. The second algorithm takes words from dictionaries and calculates the number of operations (edit distance) to create a word with error. The reason why we are using the second algorithm is the expectation that it is faster than the first one at the moment when there are not many words in the dictionaries. Therefore, we performed a test in which we try to find out the limit when the second algorithm is faster and, conversely, when the second algorithm starts to be slower than the first algorithm.

The result of the test is shown in Fig. 2. The first algorithm’s time depends only on the word’s length and is shown by a line with circle points. The other lines show the second algorithm’s time according to the number of words with the same length. It can be seen that the time complexity of the first algorithm is approximately the same as the complexity of the second algorithm when there are 25000 words with a similar length as the analyzed word in the dictionary.

Based on the test, we chose a 25000 word limit to determine which algorithm to use. Because the method searches for candidates up to edit distance 2, it is possible to count only words that are not shorter or longer by two characters than the detected word with an error. When processing a word with a typographical error of length N, the tool determines whether the number of words in the dictionary of length N-2 to N+2 is less than or greater than 25000.

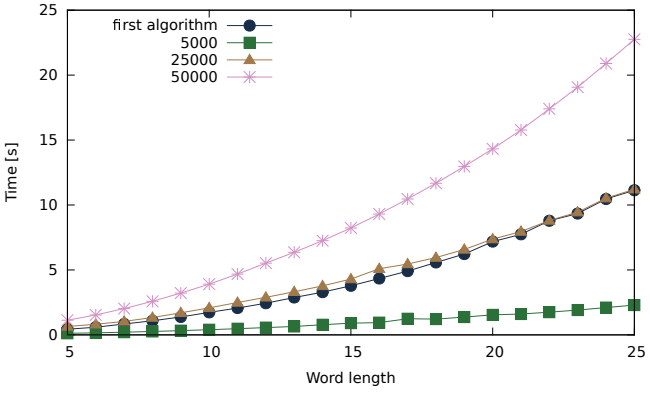

Fig. 2. Comparison of the generation time of candidates of the first and second algorithm.

3) Candidate ranking: The algorithm for candidate ranking works in two stages: assigning a base score and adjusting it based on the error type. The first stage is simple. If a word was found in the learned part of the dictionary, a score of 25 is used. Otherwise, a score of 20 is used for words from the predefined part. Afterward, it is checked whether the word is of the type domain name, string, or username and whether the candidate has the same phonetic sound as the typographical error (according to the Metaphone algorithm). If the phonetic sound is the same, the value 3 is added to the score.

In the second stage, the method takes the operations needed to change the word with an error to the correct word and calculates the penalty that will be reduced from the score. The penalty has a default value of 10. A list of conditions defining more probable errors is checked, and for each fulfilled condition, the penalty is reduced by 1. The system checks the following conditions that define more probable errors:

- inserted character is the same as the adjacent character;
- inserted character is in the fat finger distance as the adjacent character;
- changed character is in the fat finger distance as the original character;
- changed character is on the same position in different keyboard layouts (qwerty-qwertz);
- mistake is not on the first character;
- error is around the separator (dot, hyphen or underscore);
- inserted character is same as the both adjacent characters;
- inserted character is in prefix or suffix of the word;
- replaced character is visually similar to the original character;
- removed character is same as the adjacent character;
- both the original and replaced character are different, but both are the same as one of the adjacent character.

Before substituting the penalty from the score, the penalty is multiplied by the coefficient based on the operation type: insert = 0.6, replace = 0.3, delete = 0.5, transpose = 0.7).

There is one exception for which the previous calculation is not used. When a word with typographical error is capitalized (caps-lock was enabled on the user's keyboard). This operation is not a basic operation used in the edit-distance calculation, and therefore calculating scores based on these operations does not make sense. In this case, the algorithm assigns the maximum score of 25 to all capitalized words. This approach will make these words always the best candidates.

D. Example

The outputs from the individual tool parts can be seen in Fig. 3, which also shows the whole method's main idea. First, words from the converter belonging to the category domain name are displayed, which are searched for in dictionaries. The domain name "amzon.com" is not in the dictionaries and is considered a typographical error. All possible words with a maximum editing distance of 2 that can be created from the domain name are generated. All these words are searched for in dictionaries, and only six are considered as correction candidates. Candidates are ranked, and the one with the best score (amazon.com) is selected as a replacement for the original misspelled word.

Fig. 3. Output's example of individual parts of the architecture.

VI. Evaluation

This section shows that the presented diagnostic method can detect typing errors caused by end-users. The first test shows that the created heuristic for evaluating candidates sorts the individual errors as expected. The second test tests the success rate of correcting typographical errors on the created dataset. In the third test, the speed of the whole tool is measured. The last test tried to correct typographical errors in IP addresses and port numbers.

A. Ranking of candidates

The first test aims to verify the correctness of the created heuristic for scoring correction candidates. We have manually applied several errors to the word "google.com", and a score heuristic algorithm was run over the words with errors. The aim is not to verify the exact values of the calculated score but to determine whether the order of the candidates according to the calculated score adequately reflects the probability of the type of typographical error that occurred. The higher the score, the higher the probability of the candidate should be. For example, one inserted character from a fat finger distance should have a better rating (higher score) than deleting two characters.

The first test result is shown in Table II, containing incorrect words, a description of operations, and a calculated score. The calculated score sorts the items in the table. There is no exact way to determine whether the created order is correct or not. However, according to intuition, it can be stated that more probable errors compared to the words with the lowest rating are placed in the first rows.

B. Candidates proposing accuracy

The second test aims to verify that the created heuristic selects the correct candidates as a replacement for the detected errors. We have created a script that applied typing errors from regular English text to the list of domain names for this test. As a list of domain names, we have used the most visited 797641 domain names from Alexa [1]. A dataset with 2455 typographical errors from 1922 words from English Wikipedia from Roger Mitton was used to simulate real errors. Each typing error in this dataset also contains the correct form of the word.

[1] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

TABLE II
SCORING VARIOUS CANDIDATES FOR THE GOOGLE.COM DOMAIN NAME

Score	Wrong word	Operation
25.0	GOOGLE.COM	Capitalization
23.9	g0ogle.com	Substitution for a similar symbol
23.6	ggogle.com	Substitution with a duplicated symbol
23.6	giogle.com	Substitution in a fat finger distance
23.3	gwogle.com	Substitution
22.0	gogle.com	Deletion of the repeated symbol
21.8	gooogle.com	Insertion of a repeated symbol
21.5	goole.com	Deletion
21.2	giigle.com	Substitution in a fat finger distance
21.2	google.comp	Insertion at the end of the word
20.6	gooqgle.com	Insertion
19.7	ogogle.com	Transposition
18.5	gopople.com	Insertion in a fat finger distance
17.0	ggle.com	Deletion
15.2	goqoqgle.com	Insertion

The script for generating domain names with typing errors consists of four steps:

1) The list containing pairs with correct and error words has been created. E.g., the correct word ”guard” was paired with the misspelled word ”gaurd”. Only words with a maximum editing distance of 2 have been used.
2) Domain names were analyzed to determine whether they contain any correct words from the created list. E.g., the domain name *theguardian.com* contains the word ”guard”. The list of pairs was always searched in random order to prevent repeating the same type of errors.
3) If the domain name contained any valid word, this word was replaced with a misspelled word. E.g., the domain name *theguardian.com* was changed to *thegaurdian.com*.
4) After 1000 different domain names have been created with typing errors. The process has stopped.

Out of 1000 created domain names, nine domain names were such that the new word was also a valid domain name even after applying the typing error. It means that the new domain name was in the same list of domain names from Alexa, and therefore it is not possible to mark those domain names as incorrect. Of the remaining 991 words, 967 cases were correctly corrected. Four misspelled words have had more than one candidate with the best score, and one of those candidates was the correct word. The remaining 20 domain names were incorrectly repaired. For example, the ”udemy.com” domain name had a typographical error ”udemi.com”, which was corrected to ”udemu.com” because replacing the letter ”i” with the letter ”u” is more likely because it is located near to it (based on the QWERTY keyboard layout). The results are shown in Table IV, and several correctly repaired domain names with different types of errors are shown in Table III.

Another test to verify the best candidates' accuracy was made on a list of 371 usernames of employees and Ph.D. students within our faculty. All usernames are based on real names. This list was used as a dictionary to check whether the checked username is correct or not. We have randomly selected 100 usernames and applied the typing errors in the same way as in the test with domain names. From these 100 usernames with errors, 100 were correctly detected as incorrect and 99 were correctly corrected. The results are also shown in Table IV. The one incorrectly repaired username was ”ikuceran” which was created from the username ”ikucera”, however our tool repaired it as ”ikuceraj”.

TABLE III
SELECTED DOMAINS WITH ERRORS THAT WERE CORRECTLY FIXED

Wrong word	Heuristic rules	Correct word
voice-real.com	transposition; similar phonetic	voice-reel.com
panbda.tv	insert; fat finger distance	panda.tv
chasr.com	substitution; fat finger distance	chase.com
ilvoepdf.com	transposition	ilovepdf.com
weatherr.com	insert; letter duplication	weather.com
gole.co.kr	2x delete; removed repeating symbol	google.co.kr

TABLE IV
RESULT OF TESTING THE CORRECTNESS OF PROPOSED CORRECTIONS FOR DETECTED TYPOGRAPHICAL ERRORS IN DOMAIN NAMES AND USERNAMES

Data type	domains	usernames
Word count	1000	100
Detected as error	991	100
Detected as correct word	9	0
Correct best candidate	967	99
Multiple best candidates	4	0
Wrongly repaired	20	1
Success rate	97.5% (967 of 991)	99% (99 of 100)

We have detected up to 97.5% of errors in domain names and 99% of username errors in misspelled words generated from sources that contained real user typographical errors. However, it is unnecessary to achieve 100% accuracy to use the tool because even with a lower success rate, the tool can make it easier to diagnose errors. We do not consider typographical errors words that are also correct as errors. For example, if a user types ”google.cz” instead of ”google.ca”, it is impossible to detect an error because both are valid domain names. Therefore we did not count them when calculating the success rate.

C. Usage on other data types

The third test aimed to verify functionality on the domain name of IP addresses and port numbers. It turned out that the proposed method of error detection does not apply to real networks very well. There were two main reasons. The first problem was to obtain relevant data to which the tool could be applied. Most networks use DNS protocol and default port numbers, so the end-users rarely use IP addresses and ports.

The second problem was that even though some data has been collected, there is only a small difference between the individual values. Usual domain names, login names, and emails are very different and easily distinguishable. For example, login names may look like ”vecerav”, ”polcak”, or ”iletavay”. On the other hand, IP addresses inside a

single network are usually assigned from the same subnet, e.g., "147.229.176.14", "147.229.176.19," or "147.229.176.8". With these values, there is a high probability that the value with an error will also be a valid value. Even if the error will be detected, it is almost impossible to say the correct value without additional knowledge.

VII. Conclusions

This work aimed to create a method for detecting typing errors in computer network applications appearing in various data sources. The tool is intended to complement existing systems for network monitoring and troubleshooting. The tool focuses on data entered directly by end-users and then transmitted over a network or stored somewhere in application servers. An example is the configuration of the email client, which is provided manually by the user. For example, when an end-user incorrectly configures such an application and does not know why the application is not working, the end-user contacts an administrator. The administrator will use this tool which will automatically check whether the application is not working because of the typing error.

The presented method can be considered a spell-checker for network-related data. The created tool focuses only on detecting nonword errors, so if another correct word is created because of a typing error, it is not identified as an error. The tool checks for each word, whether it is present inside a dictionary, and if not, it is considered a typographical error.

At the beginning of the work, we have identified data types in which it makes sense to search for typographical errors. The most interesting data sources are network packets, NetFlow records, and log files regarding availability and amount of data containing possible user' errors. Each type of data has its advantages and disadvantages, and therefore it is impossible to say that only one specific type of data should be used.

Many kinds of typographical errors can be made when typing on the keyboard. The basic mechanism behind the emergence of such errors was identified by Damerau, who defined it in terms of insertion, transposition, replacement, and deletion of a character. Based on these operations, it is possible to measure the editing distance between two words and find similar words to be used as a correction. When there are multiple candidates for correction, it is necessary to determine the best selection. To solve this selection problem, we have developed an algorithm that considers the probabilities of individual operations' occurrence and looks for the most common types of typographical errors in candidates. Based on the highest evaluation, the best candidate for replacement is selected. In the case of equally rated candidates, there are multiple corrections offered.

The created tool works in two modes. Firstly, the learning mode is applied to create dictionaries of correct words based on the provided input data considered to be correct. Secondly, the diagnostic mode uses a learned model to detect and correct typing errors. The program can also run without the learning phase, in which case it requires a manually created model.

The tool's functionality was tested on errors in domain names and login names created by applying real-user mistakes from the English Wikipedia. Knowing the correct form of each error made it possible to determine whether the tool suggests correct correction candidates. Although the tool's success rate was less than 100% (97.5% and 99%), the tool can still speed up diagnostics by not requiring the administrator to manually analyze the data entered by the end-user.

Although the method can be applied to any data entered by a user, it is not appropriate to look for typographical errors in identifiers from layers other than from the application layer. The reason is that only the application identifiers are optimized for typing by users. There are significant differences between each value, and it is thus possible to easily estimate the intended word in the case of a typing error.

Finally, we have identified the following future works:

- The tool does not allow feedback processing of the previous errors. By marking each correction as right or wrong, the diagnostic method would prioritize previously successful candidates.
- Every word that has been seen during the learning mode is considered correct. However, when learning from data with possible errors, words that were seen only once may indicate a potential typo. It would make sense to penalize less frequent words in dictionaries. However, it would be necessary to deal with words that occur less frequently and be erroneously marked as typos, even if they are already in the dictionary.

Acknowledgment

This work was supported by the BUT FIT grant FIT-S-20-6293, "Application of AI methods to cyber security and control systems".

References

[1] Han, Y., Zhao, X. and Li, J., 2018. Computer Network Failure and Solution. Journal of Computer Hardware Engineering, 1(1).

[2] Wang, R., Wu, D., Li, Y., Yu, X., Hui, Z. and Long, K., 2012. Knight's tour-based fast fault localization mechanism in mesh optical communication networks. Photonic Network Communications, 23(2), pp.123-129.

[3] Solé, M., Muntés-Mulero, V., Rana, A.I. and Estrada, G., 2017. Survey on models and techniques for root-cause analysis. arXiv preprint arXiv:1701.08546.

[4] Orzach, Y., 2013. Network Analysis Using Wireshark Cookbook. Packt Publishing Ltd.

[5] Squicciarini, A.C., Petracca, G., Horne, W.G. and Nath, A., 2014, March. Situational awareness through reasoning on network incidents. In Proceedings of the 4th ACM conference on Data and application security and privacy (pp. 111-122).

[6] Van, T.,Tran, H. A.,Souihi, S. A. Mellouk, A. Network troubleshooting:Survey, Taxonomy and Challenges. In: IEEE.2018 International Conference onSmart Communications in Network Technologies (SaCoNeT). 2018, pp. 165–170.

[7] Zeng, H., Kazemian, P., Varghese, G. and McKeown, N., 2012. A survey on network troubleshooting. Technical Report Stanford/TR12-HPNG-061012, Stanford University, Tech. Rep.

[8] Chen, A., Wu, Y., Haeberlen, A., Zhou, W. and Loo, B.T., 2016, August. The good, the bad, and the differences: Better network diagnostics with differential provenance. In Proceedings of the 2016 ACM SIGCOMM Conference (pp. 115-128).

[9] Kögel, J., Including the Network View into Application Response Time Diagnostics using Netflow.

[10] Rudrusamy, G., Ahmad, A., Budiarto, R., Samsudin, A. and Ramadass, S., 2013. Fuzzy based diagnostics system for identifying network traffic flow anomalies. arXiv preprint arXiv:1304.7864.

[11] Qiu, T., Ge, Z., Pei, D., Wang, J. and Xu, J., 2010, November. What happened in my network: mining network events from router syslogs. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (pp. 472-484).

[12] Csikor, L. and Pezaros, D.P., 2017, December. End-host driven troubleshooting architecture for software-defined networking. In GLOBECOM 2017 IEEE Global Communications Conference (pp. 1-7). IEEE.

[13] Pullmann, J. and Macko, D., 2018, November. Network tester: A generation and evaluation of diagnostic communication in ip networks. In 2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA) (pp. 451-456). IEEE.

[14] Traverso, S., Tego, E., Kowallik, E., Raffaglio, S., Fregosi, A., Mellia, M. and Matera, F., 2014, September. Exploiting hybrid measurements for network troubleshooting. In 2014 16th International Telecommunications Network Strategy and Planning Symposium (pp. 1-6). IEEE.

[15] Kukich, K. Techniques for automatically correcting words in text. Acm Computing Surveys (CSUR). ACM. 1992, 24(4), pp.377-439.

[16] Hirst, G. "An evaluation of the contextual spelling checker of Microsoft Office Word 2007." (2008).

[17] Youssef, B. and Alwani, M.. "Ocr post-processing error correction algorithm using google online spelling suggestion." arXiv preprint arXiv:1204.0191 (2012).

[18] Chen, X., Huang, X., Mu, Y. and Wang, D., 2019, August. A Typo-Tolerant Password Authentication Scheme with Targeted Error Correction. In 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 546-553). IEEE.

[19] Chatterjee, R., Athayle, A., Akhawe, D., Juels, A. and Ristenpart, T., 2016, May. pASSWORD tYPOS and how to correct them securely. In 2016 IEEE Symposium on Security and Privacy (pp. 799-818). IEEE.

[20] Ahmad, I., Parvez, M.A. and Iqbal, A., 2019, July. TypoWriter: A Tool to Prevent Typosquatting. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 423-432). IEEE.

[21] Sun, Y.C., Tang, D.D., Zeng, Q. and Greenes, R., 2002. Identification of special patterns of numerical typographic errors increases the likelihood of finding a misplaced patient file. Journal of the American Medical Informatics Association, 9(Supplement_6), pp.S78-S79.

[22] Wang, Y.M., Beck, D., Wang, J., Verbowski, C. and Daniels, B., 2006. Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting. SRUTI, 6(31-36), pp.2-2.

[23] QasemiZadeh, B., Ilkhani, A. and Ganjeii, A., 2006, June. Adaptive language independent spell checking using intelligent traverse on a tree. In 2006 IEEE Conference on Cybernetics and Intelligent Systems (pp. 1-6). IEEE.

[24] Lianga, H.L., Watsonb, B.W. and Kourieb, D.G., Technical Report Classification for Selected Spell Checkers and Correctors.

[25] Peterson, J.L., 1980. Computer programs for detecting and correcting spelling errors. Communications of the ACM, 23(12), pp.676-687.

[26] Damerau, F.J. and Mays, E., 1989. An examination of undetected typing errors. Information Processing & Management, 25(6), pp.659-664.

[27] Bao, Z., Kimelfeld, B. and Li, Y., 2011, June. A graph approach to spelling correction in domain-centric search. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (pp. 905-914).

[28] Odell, K.M. and Russell, R.C., 1918. Soundex phonetic comparison system. US Patent, 1261167.

[29] Pollock, J.J. and Zamora, A., 1984. Automatic spelling correction in scientific and scholarly text. Communications of the ACM, 27(4), pp.358-368.

[30] Philips, L., 1990. Hanging on the metaphone. Computer Language, 7(12), pp.39-43.

[31] Elmi, M.A. and Evens, M., 1998. Spelling correction using context. In COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics.

[32] Martins, B. and Silva, M.J., 2004, October. Spelling correction for search engine queries. In International Conference on Natural Language Processing (in Spain) (pp. 372-383). Springer, Berlin, Heidelberg.

[33] Mullin, J.K. and Margoliash, D.J., 1990. A tale of three spelling checkers. Software: Practice and Experience, 20(6), pp.625-630.

[34] Aho, A.V. and Corasick, M.J., 1975. Fast pattern matching: an aid to bibliographic search. Communications of ACM, 18(6), pp.333-340.

[35] Jurafsky, D. and Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson Prentice Hall (2009).

[36] Mitton, R. English Spelling and the Computer (Studies in Language Linguistics). Addison-Wesley Longman Ltd, dec 1995. ISBN 0582234794.

[37] Levenshtein, V.I., 1966, February. Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady (Vol. 10, No. 8, pp. 707-710).

[38] Wagner, R.A. and Fischer, M.J., 1974. The string-to-string correction problem. Journal of the ACM (JACM), 21(1), pp.168-173.

[39] Hamming, R.W., 1950. Error detecting and error correcting codes. The Bell system technical journal, 29(2), pp.147-160.

[40] Pilar Angeles, M. del a Espino Gamez, A. Comparison of methods Hamming Distance, Jaro, and Monge-Elkan. In: DBKDA 2015: the seventh international conference on advances in databases, knowl-edge and data applications. 2015.

[41] Damerau, F.J., 1964. A technique for computer detection and correction of spelling errors. Communications of the ACM, 7(3), pp.171-176.

[42] Winkler, William E. "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage." (1990).

[43] Mitton., R., Birkbeck spelling error corpus, 1985. http://ota.ahds.ac.uk/ (Last accessed: March 2007).

[44] TshwaneDJe HLT. Sesotho sa leboa corpora, 2006. Private Communication.September 2006.

[45] Rudy, R. and Naga, D.S., 2018. Fast and Accurate Spelling Correction Using Trie and Damerau-levenshtein Distance Bigram. TELKOMNIKA (Telecommunication Computing Electronics and Control), 16(2), pp.827-833.

[46] Cordewener, K.A., Verhoeven, L. and Bosman, A.M., 2016. Improving spelling performance and spelling consciousness. The journal of experimental education, 84(1), pp.48-74.

[47] Van Zaanen, M. and Van Huyssteen, G., 2003. Improving a spelling checker for Afrikaans. In Computational Linguistics in the Netherlands 2002 (pp. 143-156). Brill Rodopi.

[48] Huang, Y., Murphey, Y.L. and Ge, Y., 2013, April. Automotive diagnosis typo correction using domain knowledge and machine learning. In 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM) (pp. 267-274). IEEE.

[49] Etoori, P., Chinnakotla, M. and Mamidi, R., 2018, July. Automatic spelling correction for resource-scarce languages using deep learning. In Proceedings of ACL 2018, Student Research Workshop (pp. 146-152).

[50] Carlson, A. and Fette, I., 2007, December. Memory-based context-sensitive spelling correction at web scale. In Sixth International Conference on Machine Learning and Applications (pp. 166-171). IEEE.

[51] Hussain, S. and Naseem, T., 2013. Spell checking. Crulp, Nuces, Pakistan, www. crulp. org.

[52] Dhakal, V., Feit, A.M., Kristensson, P.O. and Oulasvirta, A., 2018, April. Observations on typing from 136 million keystrokes. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-12).

[53] Rimbar, H., 2017. The Influence of Spell-checkers on Students' ability to Generate Repairs of Spelling Errors. Journal of Nusantara Studies (JONUS), 2(1), pp.1-12.

[54] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A. and Pras, A., 2014. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. IEEE Communications Surveys & Tutorials, 16(4), pp.2037-2064.

[55] Holkovič, M., Polčák, L. and Ryšavý, O., 2019, July. Application Error Detection in Networks by Protocol Behavior Model. In International Conference on E-Business and Telecommunications (pp. 3-28). Springer, Cham.

[56] Moore, T. and Edelman, B., 2010, January. Measuring the perpetrators and funders of typosquatting. In International Conference on Financial Cryptography and Data Security (pp. 175-191). Springer, Berlin, Heidelberg.

[57] Grudin, J.T., 1983. Error patterns in novice and skilled transcription typing. In Cognitive aspects of skilled typewriting (pp. 121-143). Springer, New York, NY.