Scientific Research Publishing

Open Journal of Applied Sciences, 2019, 9, 819-840
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

A Spell Checking Web Service API for Smart City Communication Platforms

Vita S. Barletta[1], Danilo Caivano[1], Antonella Nannavecchia[2*], Michele Scalera[1]

[1]Department of Informatics, University of Bari, Bari, Italy
[2]Department of Economics, Lum Jean Monnet University, Casamassima, Italy
Email: *nannavecchia@lum.it

How to cite this paper: Barletta, V.S., Caivano, D., Nannavecchia, A. and Scalera, M. (2019) A Spell Checking Web Service API for Smart City Communication Platforms. *Open Journal of Applied Sciences*, **9**, 819-840.
https://doi.org/10.4236/ojapps.2019.912066

Received: November 5, 2019
Accepted: December 6, 2019
Published: December 9, 2019

Abstract

The Internet of Things becomes Internet of Everything when in the process of communication machine-to-machine also intelligent forms of communication between human and machine are involved. Cities can be viewed as a microcosm of this interconnected system where ICT and emerging technologies can be enabling factors to transform cities in Smart Cities. Cities can take great advantage by using information intelligence to achieve important public-policy goals and, in particular, by enabling network communication channels between citizens and public administrators in order to provide information and online services in real time through platform systems rather than by means of humans, using Artificial Intelligence and Natural Language Processing techniques. This work was the first step of a wider project aimed at providing a Spell Checking Web Service API for Smart City communication platforms able to automatically select, among the large availability of open source spell checking tools, the most suitable tool based on the semantic structure of the specific textual data. The system should manage an enhanced Italian Vocabulary Database, specifically implemented to support all the tools of the system. The goal of the present work was to test, through an experimental research, the feasibility of the entire project by implementing a Spell Checking Prototype System designed to manage two selected spell checking tools. Results showed that the Spell Checking Prototype System significantly improves performances by allowing the user to select the most suitable tool for the specific semantic structure of the text. The system also enables to manage the list of exceptions, which continuously enhance the Italian Vocabulary Database. The experimentation proved scientific evidence of the validity of the project aimed at implementing a Spell Checking Web Service API in order to improve the quality of natural language data to be stored or processed in Smart City NCeSDP systems, through the use of existing spell checking tools.

Keywords

Internet of Everything, Smart City, Natural Language Processing, Spell Checking Service

1. Introduction

The "Internet of Things" or "Internet of Objects" is a concept that describes the wireless and wired Things connected to the Internet for exchanging information [1]. Nevertheless, the actual value is not created by the Things but by the connection between people, process, data, and things—the "four pillars" on which the Internet of Everything (IoE) concept is built [2] [3].

In the IoE, people are able to connect to the Internet in a wide variety of ways such as through the use of devises and social networks [4]. The IoT becomes the Internet of Everything by involving intelligent and robust forms of communication between machine-to-people (M2P), machine-to-machine, people-to-machine and people-to-people (P2P) [5].

In this context, cities can be considered as microcosms of the interconnected networks that create the IoE [4]. The IoE can give great benefits to the institutions in order to achieve many public-policy goals, including economic growth, productivity, public safety and security, environmental sustainability, and delivery of government services [6]. Billions of sensors connected to the Internet can be used to manage efficiently and effectively resources in Smart Cities [7]. Cities can take significant advantage by using information intelligence to manage city issues [8]. They can enormously benefit from connecting people, process, data, and things with the aim at enhancing business communication, facilitating employment, well-being, education and healthcare between various communities of people [9]. Public-private partnership can help to improve the living conditions of the citizens, turning cities into "Smart + Connected Communities" [10].

Smart Cities could have great benefits by enabling network communication channels to facilitate the communication between general public and administrators and provide online services to the citizens. The most natural way for administrators to interact with citizens is using natural language. For this reason, the requirements of validity and authenticity of textual messages become essential [9]. The quality of data, including text quality, is crucial to derive actionable and, more importantly, accurate insights from information.

There are many tools aimed at checking for misspelling errors, specialized for different types of natural language data. These tools are designed to identify misspelled words and notify the user of the misspellings. The programs can autocorrect the word or let the user select from potential corrections. These tools work supported by vocabularies in multiple languages and they have great results in the English language. Moreover, each tool shows more or less high performance depending on the semantic structure and the type of textual data [11]

[12] [13] [14] [15].

This work arose in the context of a wider project aimed at providing a novel application through a Spell Checking Web Service API for Smart City Network Communication and e-Services Delivery Platforms. The API service should be able to automatically select among the large availability of open source spell checking tools, the most suitable tool based on the semantic structure of the specific textual data. The system should be supported by an enhanced Italian Vocabulary Database, unique and shared by all the tools, containing a set of words considered to be correct.

In the first step of the project, object of this work, an experimental research was carried out to test the feasibility of the entire project. A Spell Checking Prototype System was implemented to manage, in this phase, only two selected spell checking tools. The system was designed to manage a list of exceptions feeding the Italian vocabulary for the selected tool in order to implement a unique Italian Vocabulary Database that should support, in the next steps, the whole system.

The paper is organized as follows: in Section 2 the importance of Network Communication and e-Services Delivery Platforms for Smart Cities was outlined; in Section 3 the entire project aimed at providing a Spell Checking Web Service API for Smart City platforms was presented; in Section 4 and 5, respectively, a brief description of Data Quality Management and Natural Language Processing techniques was proposed; in Section 6 a review of the most widespread spell checking tools was presented; in Section 7 the Spell Checking Prototype System implemented for the experimentation was described and, finally, in Sections 8 and 9, respectively, results and conclusions were presented.

2. Network Communication and E-Services Delivery Platforms for Smart Cities

The existence of communication channels between citizens and public administration is a central issue for Smart Cities. The use of advanced technologies can facilitate and stimulate citizens' involvement and participation in city life. The possibility to access to a great range of social and public services online, or e-services, through a wide range of communication channels and via multiple devices, such as mobile ones, is crucial for Smart Governments [16].

A great advantage for Smart Cities could be to enable Network Communication and e-Services Delivery Platforms (NCeSDP) to give citizens easily access to information and services in real time without the presence of a human but only using the emerging technologies. NCeSDPs could offer to citizens a great range of e-services such as municipal services, city's administrative services, health and social services, education and culture services; document requirements; information; payment of bill and fees, etc. [16]. A citizen could visit the platform to ask specific information or to access to particular services.

The implementation of NCeSDPs can only be realized based on ICT infrastructure able to manage data flows and interact in real time with citizens. The

ICT and the related emerging technologies such as IoT and IoE can be key enabling factors to transform cities into Smart Cities. The increasing development of the IoT and of the IoE is leading to the implementation of systems where all components can interact with each other in real time through the Internet. The use of AI to understand how humans interact in the social context [10] and a better customer habit analysis could play a critical role in the context of Smart Cities. It could enable more efficient administration in order to deliver better services and experiences [8] in all domains such as pollution control, traffic management, and utility services [10]. Psychology and neuroscience of human behavior can help to understand how humans achieve knowledge and learn to adapt to changing contexts [17].

In this scenario, new paradigms have come forward: the Internet of Conversational Things (IoCT) that arises from the ability of the Things to communicate and cooperate in order to create new services [1]; the Social Internet of Things (SIoT) [18] that can be viewed as an ecosystem in which people and connected devices can interact in a social structure of relationships [19]. In this framework, applications and services can be provided in a user-friendly manner relying on web technologies [20].

Communication is not only a fundamental need of human nature but it is also important in terms of commercial transactions and access to online services making their everyday tasks easier and, hence, saving time and money [21]. Communication was made more efficient and affordable by the advancement and the expansion of new technologies [21]. The increasing desire to verbally interact with devices and applications [22] and the growing preference for conversational search lead to the introduction of Conversational Agents in more and more domains. Systems able to interact with people not just by reacting schematically to a request but through dialog or conversation were implemented [1]. Conversational Agents can be implemented in order to provide a personal assistance answering to the users in real time [23] and making interaction more natural and effortless [1]. The use of natural language is a simple vehicle to realize an interoperability system that is universal and embraces people, devices and applications [24]. Speech is universal and widespread, natural and faster. These features together with the improvement in Voice Computing (VC) that include Natural Language Processing (NLP), Speech Recognition, as well as growth in Artificial Intelligence point out that the future interaction in the IoCT ecosystem will be based on speech [1].

Conversational Agents can act as an interface allowing people to access to information and services in real time through platform systems rather than by means of humans [22]. It can facilitate the human-machine interaction through the execution of different tasks such as to search answer and respond to queries based on user requests and can let users become active participants [25]. Computer programs, able to understand natural language and perform tasks, simulate conversation between humans giving the impression to the user to talk to someone else [26]. Human interaction with Things can be enhanced by using

speech since it is the most natural way to communicate [27].

3. Spell Checking Web Service API

Local governments are understanding the importance of gathering and deploying data more effectively [28]. With ever greater frequency, governments have started to utilize Iot, Big Data and computer algorithms to enhance the development and the sustainability of Smart Cities [29].

The implementation of Smart City NCeSDP systems, able to offer a wide range of information and e-services to citizens in real-time, can be conceived as APIs designed to perform multiple tasks and accessible via multiple communication channels including social networks (Facebook, Twitter, WhatsApp) and devices. The users of the NCeSDP are citizens (human-machine communication) or other machines (machine-machine communication) that need to receive particular information or e-services. Since communication between NCeSDP and citizens occurs in natural language, a Conversational Agent implemented in the NCeSDP system can be able to provide personal assistance to citizens. The Conversational Agent could interact with them in a colloquial way using natural language. These systems rely on the concepts of Artificial Intelligence, Natural Language Processing (NLP), Machine Learning (ML), and Speech Recognition techniques [30] [31]. The NLP allows retrieving facts from users' search queries or other natural language interactions with services [20].

Moreover, the machine-machine communication and the integration of objects, services and people increase the quantity and the variety of data to deal with [20]. Smart Cities use, in fact, data coming from heterogeneous sources, including various types of natural language data.

In this context, one of the most relevant issues in the implementation of Smart City NCeSDPs is the quality of data from different sources and their integration in an application to extract higher-level information and/or to provide actionable information to other services and applications [32].

The ultimate goal of the project is to provide a Spell Checking Web Service for NCeSDPs, available through API, able to check for possible misspelling errors in natural language data. The service should select, among the available spell checking tools complying with defined criteria, the most performing tool based on the specific semantic structure of the text string in order to check it and return the correct text. There is a great availability of open source tools for misspelling errors checking and they give great results in the English language. Many studies have been conducted in order to improve performance in other languages [33] [34] [35] [36].

The project is structured in the following steps:

1) in the first step, that is the object of this work, the behavior of the most widespread open source spell checking tools available has been analyzed. A Spell Checking Prototype System has been implemented in order to manage, in this phase, only two tools identified for the experimentation. The system has been

designed to manage a list of exceptions at the user's discretion. The list contains the words that must not be reported as errors and feeds the Italian vocabulary for the selected tool with the purpose to implement the Italian Vocabulary Database. The ultimate goal of this step is to test if the proposed Spell Checking Web Service API, supported by a unique and shared Italian Vocabulary Database, can work as a service for Smart City NCeSDP systems;

2) in the second step, object of future works, the Spell Checking Prototype System will use a larger but always fixed number of spell checking tools complying with defined criteria. An algorithm will select the most suitable tool based on the semantic structure of the textual data. As in the previous step, the system will be able to manage a list of exception that feeds the Italian vocabulary for each selected tool with the same purpose to enhance the Italian Vocabulary Database;

3) in the third step, the entire project will be completed with the implementation of the Spell Checking Web Service API (**Figure 1**). The system will operate an autonomous process of Data Quality Cleaning through an algorithm for selecting the most suitable tool based on the semantic structure of textual data (**Figure 1**—step 1). The system will be supported by the Italian Vocabulary Database (**Figure 1**—step 2) specifically implemented in the previous steps, that will be unique and shared by all the tools. The system will always allow the user to manage possible exceptions continuing to feed the database. The Italian Vocabulary Database will be used to support any tool connected to the system through an algorithm able to transform the database into a file with the specific features required by the selected tool (**Figure 1**—step 3 - 4). The system will use connectors to send the text and the adjusted database (**Figure 1**—step 5 - 6) to the selected tool which returns the corrected text (**Figure 1**—step 7 - 8).

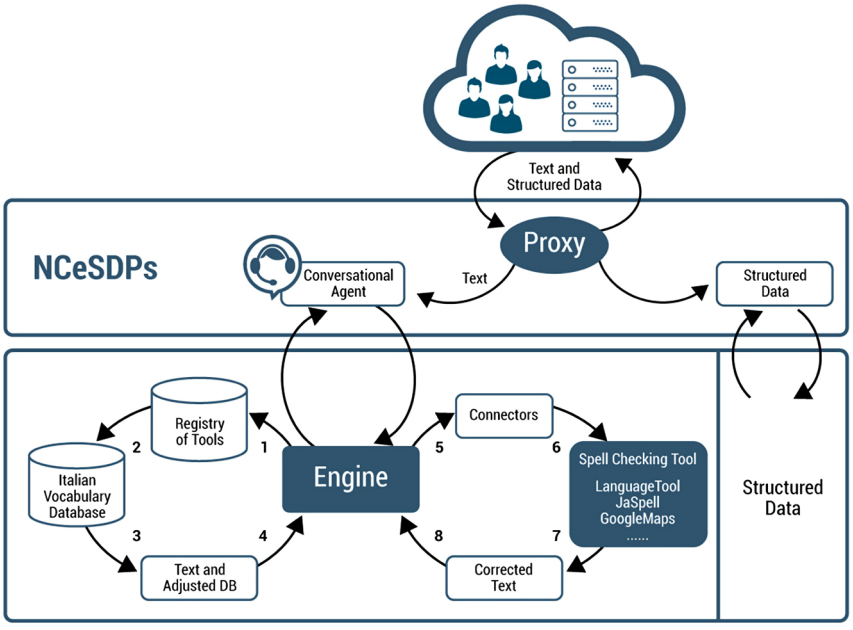

Figure 1. Spell checking web service API for smart city NCeSDPs design (SERLAB elaboration).

The experimental research presented in the work was conducted with the aim to test the feasibility of the entire project to develop a Spell Checking Web Service API for Smart City NCeSDPs by implementing a Spell Checking Prototype System as described in the first step.

4. Data Quality Management

Nowadays, public and private organizations understand the value of the data. Data are a key aspect to improve efficiency and effectiveness in everyday dynamics. However, "as organizations begin to create integrated data warehouses for decision support, the resulting Data Quality (DQ) problems become painfully clear" [37].

Data can be defined of quality if they correctly represent the aspects of the real world to which they refer to, or if they "fit for intended use in operations, decision making and planning" [38]. "Data are dirty if the user or application ends up with a wrong result or if they are able to derive some result inherent problems with the data" [39]. "Dirty data in a database routinely generate misleading or biased analytical results and decisions, and lead to loss of revenues, credibility and customers" [40].

The International Organization for Standardization (ISO), has defined two standards, ISO:25012 and ISO:25024, which define a data quality model and suggest how to measure data quality. The ISO international standards bring benefits in almost every imaginable sector standards supporting technology and ensuring the expected quality.

In order to ensure the quality of data, it is necessary to design and implement a four-step process:

1) *Data Source Definition*, which in the present research allowed the identification of the attribute to be submitted to the Data Quality process.

2) *Data Analysis*, which carries out the Data Quality Assurance operation, aimed at identifying incorrect data and inconsistencies through a data profiling approach.

3) *Verification*, which carries out the Data Quality Control to verify the goodness of the achieved results.

4) *Backflow of Cleaned Data*, which upon confirmation of the Data Base Administrator performs the correction of incorrect data with those finally corrected.

In the work, the Data Quality Assurance step was implemented through the data profiling approach. It was aimed at discovering inconsistencies and anomalies in the data for subsequent data cleansing operations as well as "examining the data available from an existing information source (e.g. a database or a file) and collecting statistics or informative summaries about the data" [41].

Obviously, the Data Analysis fundamental step is the data cleansing process. "Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database

and refers to identify incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replace, modify, or delete the dirty or coarse data" [42]. For existing datasets, the most logical solution to this problem is to clean the data, exploring the dataset for possible errors.

In the present work, the data cleansing step to ensure text quality is implemented with algorithms based on Natural Language Processing (NLP).

The Verification step is aimed at checking the adequacy of the data, with known quality, for an application, a process or a specific purpose. Generally, this step uses the information coming from the Data Analysis to decide whether to use the data to perform an analysis, to use them in an application or in a business process. For example, if the Data Quality Control finds that the data contains too many errors or inconsistencies for a specific purpose, then they will not be used, thus avoiding generating an incorrect output. High quality data must meet a set of quality criteria that include:

- *Validity*: validity is the extent to which a concept, conclusion or measurement corresponds accurately to the real world.
- *Accuracy*: the degree of closeness of the quantity to that quantity's true value.
- *Completeness*: it indicates if the available data are sufficient to meet current and future business needs.
- *Consistency*: a datum is consistent if it does not contain a contradiction.
- *Uniformity*: uniformity in the measurement specifies the unit of measurement defined and adopted by convention or by law, used as a standard for measurement of the same kind of quantity.

5. Natural Language Processing

The Natural Language Processing deals with the interactions between computers and human (natural) languages and allows to process large data in natural language. The success of any NLP application is based on the quality of the underlying textual data analyzed. NLP data quality assessment has become an important need for NLP datasets (NLP Data Cleansing Based on Linguistic Ontology Constraints).

Until the 1980s, many of the NLP systems were based on a complex set of manually written rules. Starting from the early 1980s, thanks to an NLP revolution [43], machine learning algorithms were introduced to process natural language. A new revolution in this area between the 1980s and 1990s [44] introduced statistics in machine learning. Machine learning was based on rules, more or less fixed, that did not change with the variation of natural language. The new approach to machine learning has involved the use of statistics, to automatically learn these rules, analyzing a huge set of documents written in natural language.

Different types of machine learning algorithms have been developed and applied to NLP. These algorithms take as input a set of documents from which a whole series of features are extracted. Furthermore, they use a probabilistic model based on a weighted value that is associated with each feature. These models have the advantage of being able to express a certain reliability over mul-

tiple possible answers, thus producing more reliable results.

One of the techniques exploited in NLP is string-matching. String-matching algorithms, or string comparison algorithms, are a class of string algorithms whose goal is to find all occurrences of a given pattern as a substring of another longer string [45]. Over the years, string-matching has been widely used to search in large sets of strings and to find errors and solutions in strings.

The experimental research carried out in this study has been aimed at developing a Natural Language Processing system capable of performing spell checks on textual strings stored in a structured form, thus identifying errors in the data. The Spell Checking Prototype System must be able to allow a detailed analysis of the results as well as a more general one. Furthermore, the system must enable the user to consult in detail each proposed correction and the permitted exceptions, allowing the management of a list of words that must not be reported as errors, at the user's discretion. In literature, these problems have been partially addressed by researchers and the algorithms useful for the purpose can be classified in three main categories:

- phonetic algorithms: which encode the word based on how it is pronounced;
- token based algorithms: which divide the word into tokens of variable length;
- comparison based algorithms: which define one or more elementary actions necessary to pass from string A to string B.

This work focuses on the possibility of performing spell checks on strings, giving the user the possibility to select the tool to implement the analysis. Three areas of interest in relation to the semantic structure of the textual data were identified. However, the attention was mainly focused on free text that presents a higher probability of error. In the experimentation, approximately four thousand strings from reliable sources that should not have contained errors were analyzed.

For the purposes of this research, related to the analysis of textual data, we'll consider only token based and comparison based algorithms.

Following is the state-of-the-art of the spell checking open source tools available based on token based and comparison based algorithms:

Token based algorithms

In a token-based algorithm, each string is divided into words or *N*-grams (substrings of consecutive letters). Two strings are the same if they have the same words or *N*-grams in common. This involves the need for a string segmentation technique, which allows the division of a string into its components. In English and in languages derived from Latin, space is considered a good approximation of delimiter. However, there are some limitations, in fact, in some cases, a word has multiple versions. For example, the English word "icebox" can become "ice-box" or "ice box". Another limitation derives from all those languages that tend to combine multiple words, such as German. In this case, the use of space as a delimiter character can cause problems.

N-gram

The *N*-gram model is a probabilistic model successfully used in a wide variety

of problems and domains such as spelling error detection and correction, information retrieval, speech and handwriting recognition [46]. An *N*-gram [47] is a continuous sequence of *N* elements from a given text sequence. The elements can be phonemes, syllables, letters, or words. When the elements are words, the *N*-gram is called *shingle*.

Given the word sequence w_1, w_2, …, w_n, the *N*-gram model, based on the Marcov assumption, approximates the probability of a word given all the previous words $P(w_n|w_1 \ldots w_{n-1})$ by using only the conditional probability of the $N-1$ preceding words $P(w_n|w_{n-N+1} \ldots w_{n-1})$. The general equation for *N*-gram approximation to the conditional probability of the next word in a sequence is [48]

$$\Pr\left(w_n \middle| w_1 \cdots w_{n-1}\right) \approx \Pr\left(w_n \middle| w_{n-N+1} \cdots w_{n-1}\right). \tag{1}$$

The probability of a complete word sentence can be approximately expressed by

$$\Pr\left(w_1 \cdots w_n\right) \approx \prod_{k=1}^{n} \Pr\left(w_k \middle| w_1 \cdots w_{k-1}\right). \tag{2}$$

Converting a sequence of elements into a set of *N*-grams through a vector representation allows to efficiently compare the sequence to another sequence. For example, both strings "abc" and "bca" generate exactly the same 2-gram "bc", despite {"ab", "bc"} is clearly different from {"bc", "ca"}. However, it is known that if two strings have a similar vector representation they are probably similar. This algorithm appears to be partly customizable [49], as it allows the user to establish the size of the single element, and returns good results for small size strings. The choice of selecting the *N* parameter in *N*-grams approximately is an important task since the large size of *N* leads to polynomial growth and its small size may shorten search time significantly [50].

Similarity between two strings s_1 and s_2 can be determined as follows [46]

$$\operatorname{sim}\left(s_1, s_2\right) \approx \frac{1}{n-N+1} \sum_{i=1}^{n-N+1} h(i), \tag{3}$$

where $h(i) = 1$ if *N*-element subsequence beginning from position i in s_1 appears in s_2, $h(i) = 0$ otherwise; $n - N + 1$ = number of *N*-element subsequence in s_1.

Trigrams

Trigrams represents a special case of the *N*-gram model where $N = 3$ and the strings are divided into triplets of characters (trigrams). In the case of trigrams, the similarity between two strings is determined by the number of triplets of characters in common in both strings.

Dice's coefficient

The Dice coefficient algorithm is a variation of the *N*-gram model. This measure takes into account the length of terms. The coefficient of Dice counts the sub-elements in common excluding duplicates elements. Therefore, it produces an index that is more accurate than the *N*-gram similarity. The coefficient value varies between zero and one. If two terms have no characters in common then the coefficient value is zero, if they are identical the coefficient value will be

one [46]. Given two strings X and Y the Dice coefficient can be defined as the ratio of the number of N-grams that are shared by two strings and the total number of N-grams in both strings [51] and can expressed as follows:

$$d(X,Y)=\frac{\left|2\times N-\text{grams}(X)\cap N-\text{grams}(Y)\right|}{\left|N-\text{grams}(X)\right|+\left|N-\text{grams}(Y)\right|}. \tag{4}$$

Jaro-Winkler's distance

The Jaro-Winkler's distance is a variant of the Jaro distance metric proposed in 1999 by William E. Winkler. Informally, the Jaro distance between two strings s_1 and s_2 is the minimum number of transpositions t of each character to transform one word into another and can be defined as follows [52]

$$d_j(s_1,s_2)=\frac{m}{3\cdot l_1}+\frac{m}{3\cdot l_2}+\frac{m-t}{3\cdot m}, \tag{5}$$

where l_1 and l_2 are the lengths (in characters) of s_1 and s_2, respectively. The value m is the number of matching characters of s_1 and s_2.

The Jaro-Winkler distance emphasizes prefix similarity between the two strings and can be defined as follows [52]

$$d_w(s_1,s_2)=d_j(s_1,s_2)+l\cdot p\cdot\left[1-d_j(s_1,s_2)\right], \tag{6}$$

where l is the length of the longest common prefix of s_1 and s_2, and p is a constant scaling factor that also controls the emphasis placed on prefix similarity. The score is normalized so that 0 represents perfect similarity and 1 represents complete dissimilarity. This algorithm produces excellent results in the case of small words, better if proper names of person [53].

Ratcliff-Obershelp algorithm

In the Ratcliff-Obershelp algorithm, the concept of matching characters is different from that of Jaro-Winkler distance. In particular, the algorithm finds the longest substring that the two strings s_1 and s_2 have in common. This substring is called *anchor* [53]. The Ratcliff-Obershelp algorithm can be defined as

$$d_{ro}(s_1,s_2)=\frac{2\cdot k_m}{|s_1|+|s_2|}. \tag{7}$$

The k_m value is increased by the length of the anchor. The remaining parts of the string, to the left and to the right of the anchor, are examined as if they were new strings (the first step is repeated). The process is repeated until all the characters of the two strings are analyzed. This algorithm, based on Common Longest Subsequence, is independent from the concept of language and is therefore the ideal choice if the input set is multilingual [53].

Comparison based algorithms

Comparison-based algorithms define one or more elementary operations to be performed to transform a string A into a string B. Hence, they return a number, which represents the number of elementary operations required.

Hamming distance

The Hamming distance between two strings of equal length is the number of

positions in which corresponding symbols are different. The Hamming distance between vectors X and Y is the number of positions where the symbols x_i and y_i differ and can be defined as follows

$$d_H(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i), \tag{8}$$

where $\delta(x_i, y_i) = \begin{cases} 0 & x_i = y_i \\ 1 & x_i \neq y_i \end{cases}$.

In other words, the Hamming distance measures the number of substitutions needed to convert one string into the other, or, otherwise, the minimum number of errors that may have led to the transformation of one string into the other. This algorithm is independent of the concept of language. However, the necessary condition for its application is that the two strings must have the same length.

Levenshtein distance

The Levenshtein distance is a mathematical measure for calculating the difference between two strings [54]. Informally, the Levenshtein distance between two strings a and b is the minimum number of elementary changes that allow to transform a string a into a string b. By elementary changes, it is meant the deletion of a character, the substitution of one character with another, or the insertion of a character. The Levenshtein distance between the first i characters of a and the first j characters of b can be defined as follows

$$\operatorname{dist}_{a,b}(i,j) = \begin{cases} 0 & i = j = 0 \\ i & j = 0 \text{ and } i > 0 \\ j & i = 0 \text{ and } j > 0 \\ \min \begin{cases} \operatorname{dist}_{a,b}(i-1,j)+1 \\ \operatorname{dist}_{a,b}(i,j-1)+1 \\ \operatorname{dist}_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & \text{otherwise} \end{cases}, \tag{9}$$

where $l_{a_i \neq b_j}$ is the indicator function equal to 0 when $a_i \neq b_j$ and equal to 1 otherwise.

As with the Hamming distance, the Lavenshtein distance is also independent of the concept of language and, unlike the Hamming distance, it can always be applied, regardless of the length of the two strings.

6. Analyzed Tools

After a literary review aimed at analyzing different string-matching algorithms, the two spell checking tools useful for the purpose were identified, that is Language Tool and JaSpell. Language Tool and Jaspell libraries were chosen because complying with defined criteria such as having an open source license, a good community to support the project, and having already been used in other projects with excellent feedbacks. It should be noted that, the aim of this step of the project, was not to identify the most performing tool but to use two suitable tools with the described features in order to validate the prototype system. The

ultimate goal was to evaluate the applicability of the proposed Spell Checking Web Service API for Smart City NCeSDP systems.

LanguageTool

LanguageTool is a Java library for spell checking based on traditional rule-based approach. Compared with similar libraries, LanguageTool presents some advantages:

- provides grammar checks allowing the user to add custom controls as well as include/exclude grammar checks;
- generates suggestions for misspelled words;
- is available with included dictionaries.

LanguageTool's text analysis process consists of four steps:

- text is divided into sentences;
- each sentence is divided into words;
- tags are assigned to each word (for example: "automobili": name, plural; "parlai": verb, simple past);
- the analyzed text is then compared with the native rules and the rules defined by the user.

Through a specific command it is also possible to indicate to the controller which phrases must be ignored in the spelling check.

JaSpell

JaSpell is a Java library that deals with spell checking, implemented on the basis of the Ternary Search Trees (TST) proposed by Jon Bentley and Bob Sedgewick, which is a special Trie data structure where the child nodes of a standard trie are ordered as a binary search tree. Search times in this structure are $O(log(n) + k)$ in the worst case, where n is the number of strings in the tree and k is the length of the string being searched for [55]. A TST data structure provides a fast and flexible approach to analyze the dictionary—it finds all the keys that have a given prefix, suffix, infix, or those keys that mostly match with a given pattern. The user can easily analyze the partial match tree and implement proximity functions, which allow suggesting alternatives for incorrectly written words. The Double Metaphone algorithm, developed by Lawrence Phillips [56], which combines the word frequency with the phonetic coding, is used to classify the possible corrections of a wrong word. Dictionaries are simple text files, where each line contains a word and the corresponding word frequency.

Other tools have not been used, like:

- *Cogito API* because it requires the manual insertion of ontologies, that is an extremely expensive task, and because in the case of short sentences it has difficulty delimiting the domain;
- *Amazon Lex* because it is still in preview, moreover the Italian support is still lacking and it is still little tested;
- *IBM Watson* because of its difficulty in managing fallbacks (for the Italian language), moreover the management of the execution logic through API is cumbersome, the transfer of intents and examples between different applications is not immediate; furthermore, it is a tool with a commercial use license

and therefore it was beyond our objectives;

- *Google Address* because it is used to validate addresses and, therefore, not appropriated in our experimentation since the reference domain does not include addresses.

7. The Spell Checking Prototype System

The Spell Checking Prototype System was specifically created for the experimental phase with the use of JavaFX. JavaFX is a family of application software, based on the Java platform, for creating web applications, computer applications, applications for portable devices and other types of platforms. JavaFX includes, in addition to a growing library of graphic functionalities, a real Java independent programming language, called "JavaFX script", a declarative and typed scripting language oriented to graphic programming, which makes the programming of graphic applications particularly easy. The JavaFX library is distributed with the Java Runtime Environment.

The proposed system consists of three modules:

- *settings*: to access MySql;
- *list of exceptions management*: to avoid that words not present in the dictionary of the libraries are reported as errors;
- *dashboard*: which allows to consult the graphs produced on the results and allows the download of the reports.

The experiment was conducted on the NACE Revision 2 document defined at European level, which formalizes the statistical classification of economic activities in the European Community and has about 4000 strings. The version used for the experimental phase was the Italian version showing the definition of each single existing activity with relative code issued by NACE.

The experiment was conducted in laboratory and consisted in three steps:

1) comparison between the LanguageTool and JaSpell libraries in order to identify the most suitable one for the purposes of the experimentation;

2) identification and removal of "false positives", that is, all the correct terms identified as errors because not included in the dictionary of the adopted library;

3) execution of the experiment.

Step 1: *Comparison of the libraries*

The Spell Checking Prototype System was designed to give to the user the possibility to select one of the two available libraries, LanguageTool and JaSpell.

This step involved:

- the tuning of the LanguageTool and Jaspell libraries. The database was populated with the data contained in the NACE document, consisting of a table with two fields, the activity code and the description;
- the control of the sentences contained in the database using the dictionaries and the basic rules of the LanguageTool and JaSpell libraries.

Therefore, the following parameters have been detected:

- coverage index of the sentences to be analyzed;

- number of errors detected.

The comparison between the two libraries revealed a significant better performance of the LanguageTool library on the JaSpell library.

In Table 1 are shown performance metrics related to both libraries in order to better highlight results. The *accuracy* of the LanguageTool library, 98.49%, is clearly higher than the accuracy of the JaSpell library, 90.11%. A highly significant result is stressed by *precision* which represent the ratio of correctly detected errors to the total of detected errors and shows a very low value for JaSpell equal to 13.97% and a considerably better value for LanguageTool equal to 59.86%. This result is due to the large number of false positives that are not correctly detected errors for JaSpell (9.16%) against a significant lower number of false positives for LanguageTool (1.49%). Most of the false positives reported by JaSpell referred to accented words and, in some cases, the words preceded by the article with the apostrophe, very common in the Italian language. This represents a limit for JaSpell, as they were reported as errors, negatively impacting on the quality of the analysis. Undetected errors were mostly due to spacing errors (for example, space immediately after comma), which JaSpell did not identify unlike LanguageTool. Also *recall*, which represents the ratio of the correctly detected errors to the total of actual errors including not detecting ones, presents higher value for LanguageTool equal to 98.84% compared to the 67.06% of JaSpell. The *F-measure*, which represents a weighted average of precision and recall, presents a significance difference equal to 74.56% for LanguageTool and only to 23.12% for JaSpell.

The results of this objective analysis pointed out that LanguageTool has a better performance with the Italian free text. For this reason the following steps of the experiment were implemented using the LanguageTool library.

Step 2: *Identification and removal of* "*false positives*"

This step involved the alignment of the terms contained in the NACE rev 2 document with the terms included in the Italian vocabulary of the selected LanguageTool library. This step required that for each word identified as non-existent a Google search was carried out to verify the actual existence or non-existence of the same. If the word existed, that is in case of false positives, the same was added to the list of exceptions developed for the library.

This step required three executions, with subsequent refinement obtained by iterations of the execution, before being able to declare the LanguageTool library actually ready for the step 3 of the experimentation.

Table 1. Performance metrics related to LanguageTool and JaSpell libraries.

	LanguageTool	JaSpell
Accuracy	98.49	90.11
Precision	59.86	13.97
Recall (sensitivity)	98.84	67.06
F-measure	74.56	23.12

Step 3: *Execution of the experiment*

Given as input the Nace rev 2 document, the execution of the data quality control with the LanguageTool library amended in the step 2 of the experiment, has detected a series of errors of different types present in the document; that is, tabulation, typing and syntactic errors. Tabulation errors are those relating to missing or exceeding spaces and missing or exceeding punctuation, such as those highlighted in Figure 2.

Typing errors are those related to letters incorrectly inserted or missing in the text, such as the one shown in Figure 3.

Figure 4 shows a syntactical error highlighted in the experimentation.

Taking into account that in the experimental phase the database was stored on a server of the LAN laboratory, the system had a run-time phase of 59 seconds.

8. Results

As outlined in the first step of the experiment, the comparison between the two spell checking libraries, revealed a better performance of the LanguageTool library on the JaSpell library in correctly detecting both sentences with errors and sentences without errors.

The implementation of the Spell Checking Prototype System and the refinement obtained by running subsequent iterations has shown even better results than the LanguageTool library. The system allows, in fact, managing a list of exceptions in order to enhance the Italian Vocabulary Database that supports the library, reducing the possibility to detect errors incorrectly.

Performance metrics are shown in Table 2 with the purpose to compare the Spell Checking Prototype System and the LanguageTool library and to highlight more insight about the accuracy of the results.

The LanguageTool library shows *accuracy* equal to 98.49%, increased to 99.92% with the use of the Spell Checking Prototype System supported by the Italian Vocabulary Database enhanced by managing the list of exceptions. The

Activity code	Unit of Measure	Italian Description
28993953	p/st	Altre macchine ed apparecchi per agglomerare,formare o modellare terre, pietre, gesso, cemento e altre materie minerali in polvere o inpasta (per fonderie, edilizia industria mineraria ecc.)

Figure 2. Misspelling tabulation errors.

Activity code	Unit of measure	Italian Description
10721990	T	Altri prodotti da forno senza aggiunta di zucceri o altri dolcificanti (quiches, crepes, pancakes, pizza, ecc.)

Figure 3. Misspelling typing errors.

Activity code	Unit ofmeasure	Italian Description
25735070	p/st	Forme per gomma o materie plastiche per formare ad inieizione o per compressione

Figure 4. Misspelling syntactical errors.

Table 2. Performance metrics related to spell checking prototype system and LanguageTool library.

	Spell Checking Prototype System	LanguageTool
Accuracy	99.92	98.49
Precision	97.70	59.86
Recall (sensitivity)	98.84	98.84
F-measure	98.27	74.56

most significant result is pointed out by the *precision* which increases from 59.86% of the LanguageTool library to 97.70% of the Spell Checking Prototype System. This result is due to the considerable reduction of false positives that are not correctly detected errors, from 1.49% of the LanguageTool library to 0.05% of the Spell Checking Prototype System. *Recall* remains unchanged, equal to 98.84%, and the *F-measure* increases from 74.56% of the LanguageTool library to 98.27% of the Spell Checking Prototype System.

It's clear that the Spell Checking Prototype System significantly improves performances by allowing the user to select the most suitable tool for the specific semantic structure of the text among the two available for the purposes of the experiment. Enabling to manage the list of exceptions, which continuously enhance the Italian Vocabulary Database, the Spell Checking Prototype System drastically reduces the percentage of false positives, bringing it to almost zero.

The experimentation carried out in this work proved scientific evidence of the validity of the ultimate goal of the project aimed at implementing a Spell Checking Web Service API. The Service should improve, through the use of existing spell checking tools, the quality of natural language data to be stored or processed in Smart City NCeSDP systems. The two following steps of the project, that will be object of future researches, will focus on the design and implementation of the Spell Checking Web Service API. The Service will allow managing a larger number of available spell checking tools, complying with defined criteria. Moreover, it will be able to automatically select the most suitable tool for a specific semantic structure of textual data. The Spell Checking Web Service API will be supported by a unique and shared Italian Vocabulary Database continuously enhanced by managing the list of exceptions. The Service will interactively respond to requests and will have the ability to adapt to changing contexts whose contours will be defined with careful analysis.

9. Conclusions

This study was conducted in the context of a wider project aimed at implementing a Spell Checking Web Service API for Smart City NCeSDP systems. The Service will be able to manage a large number of available spell checking tools, complying with defined criteria, and to automatically select the most proper tool depending on the semantic structure of the specific textual data. The Service will be supported by a unique and shared Italian Vocabulary Database enhanced by

managing a list of exceptions for each tool.

In this study, a Spell Checking Prototype System was implemented in order to manage two defined spell checking tools and select the most suitable tool based on the semantic structure of the text. Moreover, the system was designed to manage a list of exceptions, that is a list of words that must not be reported as errors, at the user's discretion. The list feeds the Italian vocabulary of the selected tool reducing the possibility to detect errors incorrectly.

The obtained results showed an increased accuracy in detecting sentences without errors for the Spell Checking Prototype System rather than for the selected spell checking tool, in the case the LanguageTool library. Results also revealed a high value of precision for the Spell Checking Prototype System, which pointed out a drastic reduction of false positives that are not correctly detected errors.

Hence, these results proved scientific evidence in order to validate the project to implement the Spell Checking Web Service API for Smart City NCeSDP systems, supported by a unique and shared Italian Vocabulary Database. It will be the focus of study in our future researches.

Acknowledgements

The authors are very grateful for the collaboration received from SER&Practices—Software Engineering Research & Practices, Spin-off of the University of Bari "Aldo Moro".

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Arsovski, S., Osipyan, H., Cheok, A.D. and Muniru, I.O. (2018) Internet of Speech: A Conceptual Model. *Proceedings of the* 3*rd International Conference on Creative Media, Design and Technology*, 207-363. https://doi.org/10.2991/reka-18.2018.79

[2] Miraz, M.H., Ali, M., Excell, P.S. and Picking, R. (2015) A Review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). 2015 *Internet Technologies and Applications*, Wrexham, 8-11 September 2015, 219-224. https://doi.org/10.1109/ITechA.2015.7317398

[3] Evans, D. (2013) Why Connections (Not Things) Will Change the World.

[4] Mitchell, S., Villa, N., Stewart-Weeks, M. and Lange, A. (2013) The Internet of Everything for Cities: Connecting People, Process, Data and Things to Improve the Livability of Cities and Communities. San Jose Cisco.

[5] Cardoso, P.J.S., Monteiro, J., Pinto, N., Cruz, D. and Rodrigues, J.M.F. (2019) Application of Machine Learning Algorithms to the IoE: A Survey. In: *Harnessing the Internet of Everything* (*IoE*) *for Accelerated Innovation Opportunities*, IGI Global, Hershey, PA, 31-56. https://doi.org/10.4018/978-1-5225-7332-6.ch002

[6] Bradley, J., Reberger, C., Dixit, A., Gupta, V. and Macaulay, J. (2013) Internet of

Everything (IoE): Top 10 Insights from Cisco's IoE Value at Stake Analysis for the Public Sector. *Economic Analysis*, 1-5.

[7] Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D. (2014) Sensing as a Service Model for Smart Cities Supported by Internet of Things. *Transactions on Emerging Telecommunications Technologies*, **25**, 81-93. https://doi.org/10.1002/ett.2704

[8] Soomro, S., Miraz, M.H., Prasanth, A. and Abdullah, M. (2018) Artificial Intelligence Enabled IoT: Traffic Congestion Reduction in Smart Cities. https://doi.org/10.1049/cp.2018.1381

[9] Galitsky, B. (2018) Message Validation Pipeline for Agents of the Internet of Everything. In: 2018 *AAAI Spring Symposium Series*, 119-127.

[10] Miraz, M., Ali, M., Excell, P. and Picking, R. (2018) Internet of Nano-Things, Things and Everything: Future Growth Trends. *Future Internet*, **10**, 68. https://doi.org/10.3390/fi10080068

[11] Kukich, K. (1992) Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, **24**, 377-439. https://doi.org/10.1145/146370.146380

[12] Miłkowski, M. (2012) Translation Quality Checking in Language Tool. Corpus Data across Lang., Berlin, Bern, Bruxelles, New York, Oxford, Wien, 213-223.

[13] Mozgovoy, M. (2011) Dependency-Based Rules for Grammar Checking with Language Tool. 2011 *Federated Conference on Computer Science and Information Systems*, Szczecin, Poland, 18-21 September 2011, 209-212.

[14] Ogata, J. and Goto, M. (2005) Speech Repair: Quick Error Correction Just by Using Selection Operation for Speech Input Interfaces. *Ninth European Conference on Speech Communication and Technology*, Lisbon, Portugal, 4-8 September 2005.

[15] Sarma, A. and Palmer, D.D. (2004) Context-Based Speech Recognition Error Detection and Correction. *Proceedings of HLT-NAACL* 2004: *Short Papers*, Boston, MA, 2-7 May 2004, 85-88. https://doi.org/10.3115/1613984.1614006

[16] I.R.M. Association and Others (2018) Smart Cities and Smart Spaces: Concepts, Methodologies, Tools, and Applications. IGI Global, Hershey, PA.

[17] Lee, S.W., Prenzel, O. and Bien, Z. (2012) Applying Human Learning Principles to User-Centered IoT Systems. *Computer*, **46**, 46-52. https://doi.org/10.1109/MC.2012.426

[18] Atzori, L., Iera, A. and Morabito, G. (2014) From "Smart Objects" to "Social Objects": The Next Evolutionary Step of the Internet of Things. *IEEE Communications Magazine*, **52**, 97-105. https://doi.org/10.1109/MCOM.2014.6710070

[19] Kim, J.E., Fan, X. and Mosse, D. (2017) Empowering End Users for Social Internet of Things. *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, Pittsburgh, PA, 18-21 April 2017, 71-82. https://doi.org/10.1145/3054977.3054987

[20] Hussein, D., Han, S.N., Lee, G.M., Crespi, N. and Bertin, E. (2017) Towards a Dynamic Discovery of Smart Services in the Social Internet of Things. *Computers & Electrical Engineering*, **58**, 429-443. https://doi.org/10.1016/j.compeleceng.2016.12.008

[21] Kalyani, V.L. and Sharma, D. (2015) IoT: Machine to Machine (M2M), Device to Device (D2D) Internet of Everything (IoE) and Human to Human (H2H): Future of Communication. *Journal of Management, Engineering and Information Technology*, **2**, 17-23.

[22] Brown, F.A., Lawrence, M.G. and Morrison, V.O. (2017) Conversational Virtual

Healthcare Assistant. Google Patents.

[23] Noguera-Arnaldos, J.Á., Paredes-Valverde, M.A., Salas-Zárate, M.P., Rodriguez-Garcia, M.Á., Valencia-Garcia, R. and Ochoa, J.L. (2017) im4Things: An Ontology-Based Natural Language Interface for Controlling Devices in the Internet of Things. In: Alor-Hernández, G. and Valencia-García, R., Eds., *Current Trends on Knowledge-Based Systems. Intelligent Systems Reference Library*, Springer, Cham, 3-22. https://doi.org/10.1007/978-3-319-51905-0_1

[24] Thierry, G., Zoraida, C., David, G., Michael, M. and Debopam, B. (2019) Natural Language for an Interoperable Internet of Simple Things. 2019 *IEEE* 5*th World Forum on Internet of Things*, Limerick, Ireland, 15-18 April 2019, 474-479. https://doi.org/10.1109/WF-IoT.2019.8767215

[25] Bozkurt, A. and Göksel, N. (2018) Technology Renovates Itself: Key Concepts on Intelligent Personal Assistants (IPAs). *Proceedings of* 10*th International Conference on Education and New Learning Technologies Conference*, Palma, Spain, 2-4 July, 2018, 4291-4297. https://doi.org/10.21125/edulearn.2018.1082

[26] Mostaço, G.M., De Souza, Í.R.C., Campos, L.B. and Cugnasca, C.E. (2018) AgronomoBot: A Smart Answering Chatbot Applied to Agricultural Sensor Networks. *Proceedings of* 14*th International Conference on Precision Agriculture*, **24**, 1-13.

[27] Mehrabani, M., Bangalore, S. and Stern, B. (2015) Personalized Speech Recognition for Internet of Things. 2015 *IEEE* 2*nd World Forum on Internet of Things*, Milan, Italy, 14-16 December 2015, 369-374. https://doi.org/10.1109/WF-IoT.2015.7389082

[28] Brauneis, R. and Goodman, E.P. (2018) Algorithmic Transparency for the Smart City. *Yale Journal of Law & Technology*, **20**, 103. https://doi.org/10.31228/osf.io/fjhw8

[29] Al Nuaimi, E., Al Neyadi, H., Mohamed, N. and Al-Jaroodi, J. (2015) Applications of Big Data to Smart Cities. *Journal of Internet Services and Applications*, **6**, 25. https://doi.org/10.1186/s13174-015-0041-5

[30] Devi, S., Merchant, Z.A.M., Siddiqui, M.S. and Lobo, M. (2019) Artificial Intelligence Based Personal Assistant. *Asian Journal of Convergence in Technology*, **5**, 1-4.

[31] Kar, R. and Haldar, R. (2016) Applying Chatbots to the Internet of Things: Opportunities and Architectural Elements. arXiv Preprint arXiv1611.03799. https://doi.org/10.14569/IJACSA.2016.071119

[32] Barnaghi, P.M., Bermudez-Edo, M., Tönjes, R., *et al.* (2015) Challenges for Quality of Data in Smart Cities. *Journal of Data and Information Quality*, **6**, 1-6. https://doi.org/10.1145/2747881

[33] de Mendonça Almeida, G.A., Avanço, L., Duran, M.S., Fonseca, E.R., Nunes, M.G.V. and Aluísio, S.M. (2016) Evaluating Phonetic Spellers for User-Generated Content in Brazilian Portuguese. In: Silva, J., Ribeiro, R., Quaresma, P., Adami, A. and Branco, A., Eds., *Computational Processing of the Portuguese Language. PROPOR* 2016*. Lecture Notes in Computer Science*, Springer, Cham, 361-373. https://doi.org/10.1007/978-3-319-41552-9_37

[34] Schneider, J.M., Fernández, J.L.M. and Martinez, P. (2012) A Proof-of-Concept for Orthographic Named Entity Correction in Spanish Voice Queries. In: Nürnberger, A., Stober, S., Larsen, B. and Detyniecki, M., Eds., *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation. AMR* 2012. *Lecture Notes in Computer Science*, Springer, Cham, 181-190. https://doi.org/10.1007/978-3-319-12093-5_10

[35] Oco, N. and Borra, A. (2011) A Grammar Checker for Tagalog Using Language-

Tool. *Proceedings of the 9th Workshop on Asian Language Resources*, Chiang Mai, Thailand, 12-13 November 2011, 2-9.

[36] Attia, M., Pecina, P., Samih, Y., Shaalan, K. and Van Genabith, J. (2012) Improved Spelling Error Detection and Correction for Arabic. *Proceedings of COLING* 2012: *Posters*, , 103-112.

[37] Orr, K. (1998) Data Quality and Systems Theory. *Communications of the ACM*, **41**, 66-71. https://doi.org/10.1145/269012.269023

[38] Redman, T.C. (2008) Data Driven: Profiting from Your Most Important Business Asset. Harvard Business Press, Brighton, MA.

[39] Kim, W., Choi, B.-J., Hong, E.-K., Kim, S.-K. and Lee, D. (2003) A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery*, **7**, 81-99. https://doi.org/10.1023/A:1021564703268

[40] Fan, W.F. and Geerts, F. (2012) Foundations of Data Quality Management. *Synthesis Lectures on Data Management*, **4**, 1-217. https://doi.org/10.2200/S00439ED1V01Y201207DTM030

[41] Theodore, J. (2009) Data Profiling. In: *Encyclopedia of Database Systems*, Springer, New York.

[42] Wu, S. (2013) A Review on Coarse Warranty Data and Analysis. *Reliability Engineering & System Safety*, **114**, 1-11. https://doi.org/10.1016/j.ress.2012.12.021

[43] Jurafsky, D. and Martin, J.H. (2008) Speech and Language Processing: An Introduction to Speech Recognition, Computational Linguistics and Natural Language Processing. Prentice Hall, Upper Saddle River, NJ.

[44] Johnson, M. (2009) How the Statistical Revolution Changes (Computational) Linguistics. *Proceedings of the EACL* 2009 *Workshop on the Interaction between Linguistics and Computational Linguistics*: *Virtuous*, *Vicious or Vacuous*? Athens, Greece, 30 March 2009, 3-11. https://doi.org/10.3115/1642038.1642041

[45] Fischer, M.J. and Paterson, M.S. (1974) String-Matching and Other Products. Massachusetts Institute of Technology Cambridge Project MAC.

[46] Niewiadomski, A. and Akinwale, A. (2015) Efficient Similarity Measures for Texts Matching.

[47] Broder, A., Glassman, S., Manasse, M. and Zweig G. (1997) Syntactic Clustering of the Web. *Computer Networks and ISDN Systems*, **29**, 1157-1166. https://doi.org/10.1016/S0169-7552(97)00031-7

[48] Jurafsky, D. and Martin, J.H. (2014) Speech and Language Processing. Volume 3, Pearson, London.

[49] Cavnar, W.B., Trenkle, J.M. and Mi, A.A. (1994) N-Gram-Based Text Categorization. *Proc. 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, NV, 161-175.

[50] Niewiadomski, A. and Akinwale, A. (2013) Efficient n-Gram-Based String Matching in Electronic Testing at Programming. In: Bădică, C., Nguyen, N.T. and Brezovan, M., Eds., *Computational Collective Intelligence. Technologies and Applications. ICCCI* 2013. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 661-670. https://doi.org/10.1007/978-3-642-40495-5_66

[51] Kondrak, G. (2005) N-Gram Similarity and Distance. In: Consens, M. and Navarro, G., Eds., *String Processing and Information Retrieval. SPIRE* 2005. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 115-126. https://doi.org/10.1007/11575832_13

[52] Malakasiotis, P. and Androutsopoulos, I. (2007) Learning Textual Entailment Using

SVMs and String Similarity Measures. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Prague, Czech Republic, 28-29 June 2007, 42-47. https://doi.org/10.3115/1654536.1654547

[53] Ilyankou, I. (2014) Comparison of Jaro-Winkler and Ratcliff/Obershelp Algorithms in Spell Check. Ilya Ilyankou Computer Science.

[54] Kang, S.-S. (2015) Word Similarity Calculation by Using the Edit Distance Metrics with Consonant Normalization. *Journal of Information Processing Systems*, **11**, 573-582.

[55] Martins, B. and Silva, M.J. (2004) Spelling Correction for Search Engine Queries. In: Vicedo, J.L., Martínez-Barco, P., Muńoz, R. and Saiz Noeda, M., Eds., *Advances in Natural Language Processing. EsTAL* 2004. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 372-383. https://doi.org/10.1007/978-3-540-30228-5_33

[56] Philips, L. (2000) The Double Metaphone Search Algorithm. *C/C++ Users Journal*, **18**, 38-43.