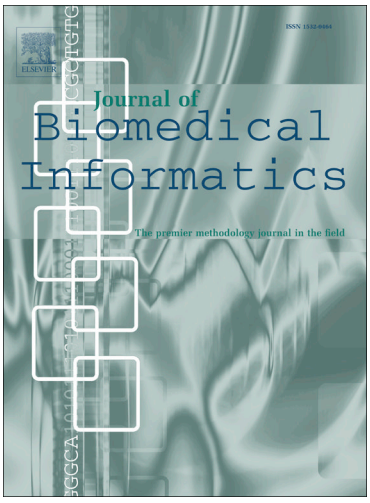


From narrative descriptions to MedDRA: automagically encoding adverse drug reactions

Carlo Combi, Margherita Zorzi, Gabriele Pozzani, Ugo Moretti, Elena Arzenton

PII: S1532-0464(18)30127-8
DOI: https://doi.org/10.1016/j.jbi.2018.07.001
Reference: YJBIN 3006

To appear in: *Journal of Biomedical Informatics*

Received Date: 28 November 2017
Accepted Date: 1 July 2018

Please cite this article as: Combi, C., Zorzi, M., Pozzani, G., Moretti, U., Arzenton, E., From narrative descriptions to MedDRA: automagically encoding adverse drug reactions, *Journal of Biomedical Informatics* (2018), doi: https://doi.org/10.1016/j.jbi.2018.07.001

From narrative descriptions to MedDRA: automagically encoding adverse drug reactions

Carlo Combi[a], Margherita Zorzi[a], Gabriele Pozzani[b], Ugo Moretti[b], Elena Arzenton[b]

[a]*Department of Computer Science, University of Verona, Italy*
[b]*Department of Diagnostics And Public Health, University of Verona, Italy*

Abstract

Context The collection of narrative spontaneous reports is an irreplaceable source for the prompt detection of suspected adverse drug reactions (ADRs). In such task qualified domain experts manually revise a huge amount of narrative descriptions and then encode texts according to `MedDRA` standard terminology. The manual annotation of narrative documents with medical terminology is a subtle and expensive task, since the number of reports is growing up day-by-day.
Objectives Natural Language Processing (NLP) applications can support the work of people responsible for pharmacovigilance. Our objective is to develop NLP algorithms and tools for the detection of ADR clinical terminology. Efficient applications can concretely improve the quality of the experts' revisions. NLP software can quickly analyze narrative texts and offer an encoding (i.e., a list of `MedDRA` terms) that the expert has to revise and validate.
Methods MagiCoder, an NLP algorithm, is proposed for the automatic encoding of free-text descriptions into `MedDRA` terms. MagiCoder procedure is efficient in terms of computational complexity. We tested MagiCoder through several experiments. In the first one, we tested it on a large dataset of about 4500 manually revised reports, by performing an automated comparison between human and MagiCoder encoding. Moreover, we tested MagiCoder on a set of about 1800 reports, manually revised ex novo by some experts of the domain, who also compared automatic solutions with the gold reference standard. We also provide two initial experiments with reports written in English, giving a first evidence of the robustness of MagiCoder w.r.t. the change of the language.
Results For the current base version of MagiCoder, we measured an average recall and precision of 86.9% and 91.8%, respectively.
Conclusions From a practical point of view, MagiCoder reduces the time required for encoding ADR reports. Pharmacologists have only to review and validate the `MedDRA` terms proposed by the application, instead of choosing the right terms among the 70K low level terms of `MedDRA`. Such improvement in the

Email addresses: `carlo.combi@univr.it` (Carlo Combi), `margherita.zorzi@univr.it` (Margherita Zorzi), `gabriele.pozzani@univr.it` (Gabriele Pozzani), `ugo.moretti@univr.it` (Ugo Moretti), `elena.arzenton@univr.it` (Elena Arzenton)

efficiency of pharmacologists' work has a relevant impact also on the quality of the subsequent data analysis. We developed MagiCoder for the Italian pharmacovigilance language. However, our proposal is based on a general approach, not depending on the considered language nor the term dictionary.

Keywords: Natural Language Processing, Healthcare informatics, Pharmacovigilance, Adverse Drug Reactions, Term identification

1. Introduction

Pharmacovigilance includes all activities aimed to systematically study risks and benefits related to the correct use of marketed drugs.

Uncommon adverse drug reactions (ADRs), such as slowly-developing pathologies (e.g., carcinogenesis) or pathologies related to specific groups of patients, are hardly discovered before the marketing. It may happen that drugs are withdrawn from the market after the detection of unexpected side effects. Thus, it stands to reason that the post-marketing control of ADRs is a necessity, considering the mass production of drugs. As a consequence, pharmacovigilance plays a crucial role in human healthcare improvement [1].

Spontaneous reporting is the main pharmacovigilance method in order to identify adverse drug reactions once drugs have been marketed [2]. Through spontaneous reporting, healthcare professionals, patients, and pharmaceutical companies can voluntarily send information about suspected ADRs to the national regulatory authority[1]. Spontaneous reports provide pharmacologists and regulatory authorities with early alerts, by considering every drug on the market and every patient category.

In last years the number of ADR reports in Italy has grown rapidly, going from approximately 6000 in 2006 to more than 50000, as shown in Figure 1.

Since the post-marketing surveillance of drugs is of paramount importance, such an increase is certainly positive. At the same time, the manual review of reports became difficult and often unbearable both by people responsible for pharmacovigilance and by regional centers. Each report must be checked, in order to control its quality; it is consequently encoded and transferred to the National Network of Pharmacovigilance (RNF).

Since 2014, the application `VigiFarmaco` helps people responsible for pharmacovigilance in the review task. An overview of `VigiFarmaco` and the Italian pharmacovigilance activities is proposed in Section 2.1. As `VigiFarmaco` simplifies the work of pharmacologists since its first release, some further improvements of the software became compelling in the light of the recent increase of reports. An essential onerous step in the validation of spontaneous reports is the encoding of the free text into the `MedDRA` (Medical Dictionary for Regulatory Activities) terminology, through which the reporter describes one or

[1] in Italy, the Drug Italian Agency AIFA – Agenzia Italiana del FArmaco, http://www.agenziafarmaco.gov.it/

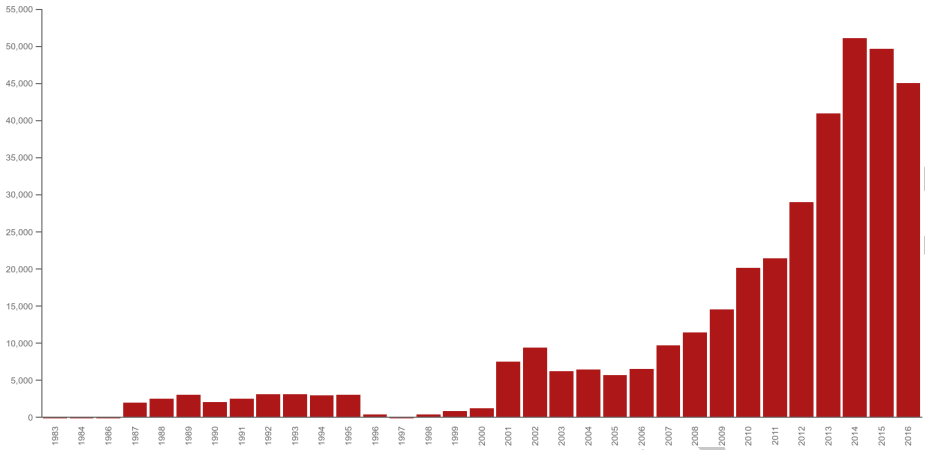


Figure 1: The yearly increasing number of reports of suspected adverse drug reactions in Italy.

more adverse drug reactions. `MedDRA` is a medical terminology introduced with the purpose of standardizing and facilitating the sharing of information about medicinal products in particular with respect to regulatory activities [3].

The description of a suspected ADR through narrative text could seem redundant/useless. Indeed, one could reasonably imagine sound solutions based either on an autocompletion form or on a menu with `MedDRA` terms. In these solutions, the description of ADRs would be directly encoded by the reporter and no expert work for `MedDRA` terminology extraction would be required. However, such solutions are not completely suited for the pharmacovigilance domain and the narrative description of ADRs remains a desirable feature, for at least two reasons. First, the description of an ADR by means of one of the seventy thousand `MedDRA` terms is a complex task. In most cases, the reporter who points out the adverse reaction is not an expert in `MedDRA` terminology. This holds in particular for consumers, but it is still valid for several professionals. Thus, describing ADRs by means of natural language sentences is simpler. Second, the choice of the suitable term(s) from a given list or from an autocompletion field can influence the reporter and limit her/his expressiveness. As a consequence, the quality of the description would be also in this case undermined. Therefore, `VigiFarmaco` offers a free-text field for specifying the ADR with all the possible details, without any restriction about the content or strict limits to the length of the written text. Consequently, `MedDRA` encoding has then to be manually implemented by qualified people responsible for pharmacovigilance, before the transmission to RNF. As this work is expensive in terms of time and attention required, a problem about the accuracy of the encoding may occur given the continuous growing of the number of reports and the number of people involved in this task (about 300 pharmacologists in Italy).

According to the described scenario, in this paper we propose MagiCoder, a natural language processing (NLP) [4] algorithm with the corresponding soft-

ware tool, which automatically assigns one or more terms from a dictionary to a narrative text. This problem is sometimes called *text normalization* in literature [5]. Since the *normalization* accomplishes a number of different shades of meaning, we will use, equivalently, the word *encoding*. MagiCoder has been first developed for supporting pharmacologists in using `VigiFarmaco`, providing them with an automatic `MedDRA` encoding of the ADR descriptions collected online, that they may correct or accept as is. In this way, the encoding task, previously completely manual, becomes semi-automatic, reducing errors and the required time for accomplishing it.

In spite of its first goal, MagiCoder has now evolved in an autonomous algorithm and software usable in all contexts where terms from a dictionary have to be recognized in a narrative text. With respect to other solutions already available in literature, MagiCoder has been designed to be efficient, less computationally expensive, and unsupervised. MagiCoder uses stemming in order to recognize singular/plural and masculine/feminine forms. Moreover, it uses string distance and other techniques to find best matching terms, discarding similar and non optimal terms. With respect to the preliminary version proposed in [6], we extended MagiCoder by following several directions, listed below.

- First, we refined the procedure: MagiCoder has been equipped with some heuristic criteria.
- MagiCoder computational complexity has been carefully studied. We will show that it is linear in the size of the dictionary (in this case, the number of Low Level Terms in `MedDRA`) and of the text description.
- We performed some accurate tests of MagiCoder performances. By means of well-known statistical measures (recall and precision), we collected a significant set of quantitative information about the effective behavior of the procedure. We refined the automatic experiment proposed in [6]. Moreover, we run a second experiment. We created a gold reference standard, involving three experts of the domain, on a significative sample of narrative reports. Pharmacologists reviewed reports and manually compared automatic and human solutions.
- We provide some evidences about MagiCoder performances on English written texts through some preliminary tests.
- We discuss some crucial key-points we met in the development of this version of the algorithm. We propose short-time solutions we are studying as work in progress, such as changes in stemming algorithm, generating and adding synonyms, and term filtering heuristics.

The paper is organized as follows. In Section 2 we provide some background notions and we discuss related work. In Section 3 we present MagiCoder, by providing both a qualitative description and the pseudocode. In Section 3.4 we describe the user interface of the related software tool. In Section 4 we explain the experiments we developed to test MagiCoder performances and we discuss

the results. Section 5 is devoted to some discussions. In Section 6 we summarize the main features of our work and sketch some future work.

2. Background and related work

In this section we first provide some background on the Italian pharmacovigilance task and the related software tool. We then introduce the `MedDRA` terminology about adverse drug reactions. Then, we discuss some related work on natural language process and mining in medicine.

2.1. The Italian pharmacovigilance and the application `VigiFarmaco`

The Italian system of pharmacovigilance requires that in each local healthcare structure (about 320 in Italy) there is a qualified person responsible for pharmacovigilance. Her/his assignment is to collect reports of suspected ADRs and to send them to the RNF (*Rete Nazionale di Farmacovigilanza*, in English *National Network of Pharmacovigilance*) within seven days since they have been received[2]. Once reports have been notified and sent to RNF, they are analysed by both local pharmacovigilance centres and by AIFA (*Agenzia Italiana del FArmaco*, in English *Italian Medicines Agency*). Subsequently, they are sent to EudraVigilance [7] and to VigiBase [8] (the European and the worldwide pharmacovigilance network RNF is part of, respectively). In general, spontaneous ADR reports are filled out by health care professionals (e.g., medical specialists, general practitioners, nurses), but also by consumers.

Recently, to increase the efficiency in collecting and managing ADR reports, a web application, called `VigiFarmaco`[3], has been designed and implemented for the Italian pharmacovigilance. Through `VigiFarmaco`, a spontaneous report can be filled out online by both healthcare professionals and consumers (through different user-friendly forms), as anonymous or registered users. The user is guided in compiling the report, since it has to be filled step-by-step (each phase corresponds to a different report section, i.e., "Patient", "Adverse Drug Reaction", "Drug Treatments", and "Reporter", respectively). At each step, data items are validated and only when all of them have been correctly inserted the report can be successfully submitted.

Once ADR reports are submitted, they need to be validated by a pharmacovigilance supervisor. `VigiFarmaco` provides support also in this phase and is useful also for pharmacovigilance supervisors. Indeed, `VigiFarmaco` reports are high-quality documents, since they are automatically validated (the presence, the format, and the consistency of data are validated at the filling time). As a consequence, they are easier to review (especially with respect to printed reports). Moreover, thanks to `VigiFarmaco`, pharmacologists can send reports (actually, XML[4] files) to RNF by simply clicking a button, after reviewing it.

[2] According to the Italian Law, Art. 132 of Legislative Decree Number 219 of 04/24/2006.
[3] Available at https://www.vigifarmaco.it
[4] eXtensible Markup Language

Online reports have grown up to become the 30% of the total number of Italian reports. As expected, it has been possible to observe that the average time between the dispatch of online reports and the insertion into RNF is sensibly shorter with respect to the insertion from printed reports. Since 2015, MagiCoder is included as `VigiFarmaco` plug-in. This extension further improves pharmacovigilance activities, offering an effective support to the (previously completely manual) free text encoding.

A partial screenshot of `VigiFarmaco` User Interface, concerning the `MedDRA` automatic encoding feature, is proposed in Section 3.4.

Since November 2017 EMA (European Medicines Agency) changed the way ADR reports are gathered and managed in the European Union. In particular, in all EU countries ADR reports are not anymore managed by the national medicines agencies and subsequently sent to EudraVigilance, but they are directly sent to EudraVigilance. EudraVigilance has then to send ADRs reports to the national medicines agencies and to VigiBase. Moreover, EudraVigilance allows all pharmaceutical companies to retrieve ADR reports related to their medicinal products.

2.2. `MedDRA` dictionary

The Medical Dictionary for Regulatory Activities, `MedDRA` [3], is a medical terminology used to classify adverse event information associated with the use of biopharmaceuticals and other medical products (e.g., medical devices and vaccines). Coding these data to a standard set of `MedDRA` terms allows health authorities and the biopharmaceutical industry to exchange and analyze data related to the safe use of medical products [9]. It has been developed by the International Conference on Harmonization (ICH); it belongs to the International Federation of Pharmaceutical Manufacturers and Associations (IFPMA); it is controlled and periodically revised by the `MedDRA` MSSO (Mainteinance And Support Service Organization). `MedDRA` is available in eleven European languages and in Chinese and Japanese too. It is updated twice a year (in March and in September), following a collaboration-based approach. Everyone can propose new reasonable updates or changes (due to effects of events as the onset of new pathologies) and a team of experts eventually decides about the publication of updates. `MedDRA` terms are organised into a hierarchy: the `SOC` (System Organ Class) level includes the most general terms; the `LLT` (Low Level Terms) level includes more specific terminologies. Between `SOC` and `LLT` there are three intermediate levels: `HLGT` (High Level Group Terms), `HLT` (High Level Terms), and `PT` (Preferred Terms).

The encoding of ADRs through `MedDRA` is extremely important for report analysis as for a prompt detection of problems related to drug-based treatments. Thanks to `MedDRA` it is possible to group similar/analogous cases described in different ways (e.g., by synonyms) or with different details/levels of abstraction.

Table 1 shows an example of the hierarchy: reaction *Itch* (`LLT`) is described starting from *Skin disorders* (`SOC`), *Epidermal conditions* (`HLGT`), *Pruritus NEC* (`HLT`), and *Pruritus* (`PT`). Preferred Terms are Low Level Terms chosen to be representative of a group of terms. It should be stressed that the hierarchy

MedDRA Level	MedDRA Term
SOC (System Organ Class)	Skin disorders
HLGT (High Level Group Terms)	Epidermal conditions
HLT (High Level Terms)	Pruritus NEC
PT (Preferred Terms)	Pruritus
LLT (Low Level Terms)	Itch

Table 1: MedDRA Hierarchy - an Example

is multiaxial: for example, a PT can be grouped into one or more HLT, but it belongs to only one primary SOC term.

2.3. Natural language processing and text mining in medicine

The automatic detection of adverse drug reactions from text has received notable attention in pharmacovigilance research. Narrative descriptions of ADRs come from heterogeneous sources: spontaneous reporting, electronic health records, clinical reports, and social media. In [10–14] some NLP approaches [15, 16] have been proposed for the extraction of ADRs from text.

In [11] Wang et al. collect narrative discharge summaries from the Clinical Information System at New York Presbyterian Hospital. The NLP system MedLEE (Medical Language Extraction and Encoding system) is applied to this collection for identifing medication events and entities, which could be potential adverse drug events. Co-occurrence statistics with adjusted volume tests were used to detect associations between the two types of entities, to calculate the strengths of the associations, and to determine their cutoff thresholds.

In [17] Toldo et al. report on the adaptation of a machine learning-based system for the identification and extraction of ADRs in case reports. The role of NLP approaches in optimised machine learning algorithms is also explored in [18], where Gonzales et al. address the problem of automatic detection of ADR assertive text segments from several sources, focusing on data posted by users on social media (Twitter and DailyStrength, a health care oriented social media). Existing methodologies for NLP are discussed and an experimental comparison between NLP-based machine learning algorithms over data sets from different sources is proposed. Moreover, the authors address the issue of data imbalance for the ADR description task.

In [19] Yang et al. propose to use association mining and Proportional Reporting Ratio (PRR, a well-known pharmacovigilance statistical index) to mine associations between drugs and adverse reactions from the user contributed content in social media. In order to extract adverse reactions from on-line text (from healthcare communities), the authors apply the Consumer Health Vocabulary[5] to generate ADR lexicon. ADR lexicon is a computerized collection

[5]Available at http://www.consumerhealthvocab.org

of health expressions derived from actual consumer utterances, linked to professional concepts and reviewed and validated by professionals and consumers. Narrative text is preprocessed following standard NLP techniques (such as stop word removal, see Section 3.1). An experiment using ten drugs and five adverse drug reactions is proposed. The Food and Drug Administration alerts are used as the gold standard, to test the performance of the proposed techniques. The authors developed algorithms to identify ADRs from threads of drugs, and implemented association mining to calculate leverage and lift for each possible pair of drugs and adverse reactions in the dataset. At the same time, PRR is also calculated.

In [20], a text extraction tool is implemented on the .NET platform for preprocessing text (removal of stop words, Porter stemming [21] and use of synonyms) and matching medical terms using permutations of words and spelling variations (Soundex, Levenshtein distance and Longest common subsequence distance [22]). Its performance has been evaluated on both manually extracted medical terms from summaries of product characteristics and unstructured adverse effect texts from Martindale [23] (a medical reference for information about drugs and medicines) using the WHO-ART (World Health Organization Adverse Reaction Terminology) and `MedDRA` medical terminologies. A lot of linguistic features have been considered and a careful analysis of performances has been provided.

In [24] Coffman et al. develop an algorithm in order to help coders in the subtle task of auto-assigning ICD-9 (International Classification of Diseases, Ninth Revision) codes to clinical narrative descriptions. Similarly to MagiCoder, input descriptions are proposed as free text. The test experiment takes into account a reasoned data set of manually annotated radiology reports, chosen to cover all coding classes according to ICD-9 hierarchy and classification: the test obtains an accuracy of 77%.

Attardi et al. use machine learning techniques to address the the problem of extracting knowledge from clinical records written in Italian by physicians in [5]. They perform recognition of relevant entities (such as symptoms, diseases, treatments, and so on), and then determine their semantic relations, and other challenging tasks such as mapping entities to a thesaurus, and identifying whether the narrative context of an expression is positive or negative. Experiments are performed on medical data provided in the context of a regional research project on technologies for health care[6] extending semi-automatically generated corpora and defining ad-hoc new sets of annotated documents.

In [25], Ribeiro et al. evaluate the retrieval performance of an algorithm that automatically categorizes medical documents assigning an International Code of Disease (ICD). The algorithm is based on well-known information retrieval techniques, operates in a fully automatic mode, and requires no supervision or training data. Performances are evaluated on a data set of about 20k documents, and authors reveal the algorithm attains levels of average precision in

[6] http://progetto-ris.isti.cnr.it/

the $70 - 80\%$ range. They also analyze some case studies of those documents whose categorization is not (partially or completely) in accordance with the one provided by the human specialists. In [26] some methods for automatically categorizing clinical documents are described and compared. In order to evaluate the automatic classification approaches, the authors exploit the PALGA data set[7], a collection of 14 million pathological reports, each of which has been classified by a domain expert.

In [27] Finite-State Transducers (FSTs, finite-state machines with a memory tape) are used to classify medical records (written in Spanish language) by their diagnostic terms, according to the International Classification of Diseases Clinical Modification (ICD-9-CM) and SNOMED CT [8]. The authors deal with the twofold challenge of treating informal natural language and performing a large-scale classification problem. FSTs were proven to be efficient in the large-scale classification task.

In [28] the natural language processor MEDSYNDIKATE is introduced. MEDSYNDIKATE couples Grammatical knowledge, semantic knowledge, and conceptual (ontological) knowledge. It automatically acquires medical information from pathology-oriented clinical reports, transferring text information into conceptual representation structures, which constitute a corresponding text knowledge base. The tool is adapted to deal with tricky text structures, such as various forms of anaphoric reference relations spanning several sentences. The authors also describe the framework for a preliminary evaluation of their approach to information extraction in terms of the semantic interpretation of a class of syntactic patterns in medical documents.

An interesting review about the role of NLP in clinical decision support systems can be found in [29]. In [30] Banda et al. deal with the standardization of medical ADR terminologies, proposing a semantically and linguistically enriched version of the US Food and Drug Administration Adverse Event Reporting System (FAERS).

Other related papers about pharmacovigilance, automatic support to diagnostic decisions, machine learning and data mining, are [31–34].

3. Materials: MagiCoder, an NLP software for ADR automatic encoding

A natural language ADR description is a completely free text. The user has no limitations and can potentially write anything. Indeed, a number of online ADR descriptions actually contains information not directly related to drug effects. Thus, an NLP software has to face and solve many issues: Trivial orthographical errors; Use of singular versus plural nouns; The so called "false positives", i.e., syntactically retrieved inappropriate results, which are closely resembling to correct solutions; The structure of the sentence, i.e., the way an

[7] http://www.palga.nl/en/public-pathology-database/
[8] http://www.snomed.org/snomed-ct

assertion is built up in a given language. Also the "intelligent" detection of linguistic connectives is a crucial issue. For example, the presence of a negation can potentially change the overall meaning of a description.

In general, a satisfactory automatic support of human reasoning and work is a subtle task. For example, the uncontrolled extension of the dictionary with auxiliary synonymous (see Section 5.2) or the naive ad-hoc management of particular cases can limit the efficiency and the desired behaviour of the algorithm. For these reasons, we carefully designed MagiCoder, even through a side-by-side collaboration between pharmacologists and computer scientists, in order to yield an efficient tool, capable to really support pharmacovigilance activities.

In literature, several NLP algorithms already exist, and several interesting approaches (such as the so-called morpho analysis of natural language) have been studied and proposed [4, 35, 36]. According to the described pharmacovigilance domain, we considered algorithms for the morpho-analysis and the part-of-speech (PoS) extraction techniques [4, 35] too powerful and general purpose for the solution of our problem. Indeed, in most cases ADR descriptions are written in a very succinct way, without using verbs, punctuation, or other lexical items, and introducing acronyms. Moreover, clinical and technical words are often not recognized correctly because not included in usual dictionaries. All these considerations limit the benefits of using morpho-analysis and PoS for our purposes.

Thus, we decided to design and develop an ad-hoc algorithm for the problem we are facing, namely that of deriving `MedDRA` terms from narrative text and mapping segments of text in effective `LLT`s. This task has to be done in a feasible time (we want that each interaction user/MagiCoder requires less than a second) and the solution offered to the expert has to be readable and useful. Therefore, we decided to ignore the *structure* of the narrative description and address the issue in a simpler way. Main features of MagiCoder can be summarized as follows:

- It requires *a single linear scan of the narrative description.* As a consequence, our solution is particularly efficient in terms of computational complexity.
- It has been designed and developed for the specific problem of mapping Italian text to `MedDRA` dictionary, but we claim the way MagiCoder has been developed is sound with respect to language and dictionary changes (see Section 6).
- The current version of MagiCoder is only based on the pure syntactical recognition of the text and it does not exploit any external synonym dictionary; in Section 3.4 we will discuss how synonyms may be used to increase MagiCoder performances. In particular, we will discuss how a naïve approach to synonyms worsen computational and retrieval performances, while we will show through experimental results and empirical observations that a prudent and suitable use of an external dictionary produces

an improvement of performances.

In this paper we consider the Italian context of Pharmacovigilance and, as a consequence, we will consider textual descriptions written in Italian language. We will discuss the potentiality of MagiCoder for other languages and some preliminary results in Sections 4.3 and 6.

3.1. MagiCoder: overview

The main idea of MagiCoder is that a single linear scan of the free text is sufficient, in order to recognize MedDRA terms.

From an abstract point of view, we try to recognize, in the narrative description, *single words* belonging to LLTs. Recognised words do not necessarily occupy consecutive positions in the text. This way, we try to "reconstruct" MedDRA terms, taking into account the fact that in a description the reporter can possibly permute or omit words. As we will show, MagiCoder has not to deal with computationally expensive tasks, such as taking into account subroutines for permutations and combinations of words (as, for example, in [20]).

We can distinguish five phases in the procedure that will be discussed in detail in Sections 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5, respectively.

1. *Definition of ad-hoc data structures.* The design of data structures is central to perform an efficient computation; our main data structures are hash tables, in order to guarantee an efficient access both to MedDRA terms and to words belonging to MedDRA terms.

2. *Preprocessing of the original text.* Tokenization (i.e., segmentation of the text into syntactical units), stemming (i.e., reduction of words to a particular root form), elimination of computationally irrelevant words.

3. *Word-by-word linear scan of the description and "voting task".* A word "votes" LLTs it belongs to. For each term voted by one or more words, we store some information about the retrieved syntactical matching.

4. *Weights calculation.* Recognized terms are weighted depending on information about syntactical matching.

5. *Sorting of voted terms and winning terms release.* The set of voted terms is pruned, terms are sorted and finally a solution (a set of winning terms) is released.

3.1.1. Definition of ad-hoc data structures

The algorithm proceeds with a word-by-word comparison. We iterate on the preprocessed text and we test if a single word w, a token, occurs into one or many LLTs.

In order to efficiently test if a token belongs to one or more LLTs, we need to know which words belong to each term. This can be done simply by indexing MedDRA LLTs words. The LLT level of MedDRA is actually a set of *phrases*, i.e., *sequences of words*. By scanning these sequences, we build a *meta-dictionary*

of all the words which compose LLTs. As we will describe in Section 3.3, in $O(mk)$ time units (where m and k are the cardinality of the set of LLTs and the length of the longest LLT in MedDRA, respectively) we build a hash table having all the words occurring in MedDRA as keys, where the value associated to key w_i contains information about the set of LLTs containing w_i. This way, we can verify the presence in MedDRA of a word w encountered in the ADR description in constant time. We call this meta-dictionary DictByWord. We build a meta dictionary also from a stemmed version of MedDRA, to verify the presence of stemmed descriptions. We call it DictByWordStem. Finally, also the MedDRA dictionary is loaded into a hash table according to LLT identifiers and, in general, all our main data structures are hash tables.

We aim to stress that, to retain efficiency, we preferred exact string matching with respect to approximate string matching, when looking for a word into the meta dictionary. Approximate string matching would allow us to retrieve terms that would be lost in exact string matching (e.g., we could recognize misspelled words in the ADR description), but it would worsen the performances of the text recognition tool, since direct access to the dictionary would not be possible. We discuss the problem of retrieving syntactical variations of the same words and the problem of addressing orthographical errors in Section 6.

3.1.2. Preprocessing of the original ADR description

A natural language ADR description has to be preprocessed in order to perform an efficient computation. We adopt a well-known technique such as tokenization [37]. A phrase is reduced to *tokens*, i.e., syntactical units which often, as in our case, correspond to words. A tokenized text can be easily manipulated as an enumerable object, e.g., an array. A *stop word* is a word that can be considered irrelevant for the text analysis (e.g., an article or an interjection). Words classified as stop-words are removed from the tokenized text. In particular, in this release of our software we decided to not take into account *connectives*, e.g., conjunctions, disjunctions, negations. The role of connectives, in particular of negation, is discussed in Section 5.

A fruitful preliminary work is the extraction of the corresponding *stemmed* version from the original tokenized and stop-word free text. Stemming is a linguistic technique that, given a word, reduces it to a particular kind of root form [21, 37]. It is useful in text analysis, in order to avoid problems such as missing word recognition due to singular/plural forms (e.g., hand/hands). In some cases, stemming procedures are able to recognize the same root both for the adjectival and the noun form of a word. Stemming is also potentially harmful, since it can generate so called "false positive" terms. A meaningful example can be found in Italian language. The plural of the word *mano* (in English, *hand*) is *mani* (in English, *hands*), and their stemmed root is *man*, which is also the stemmed version of *mania* (in English, *mania*). Several stemming algorithms exist, and their impact on the performances of MagiCoder is discussed in Section 5.

3.1.3. Word-by-word linear scan of the description and voting task

MagiCoder scans the text word-by-word (remember that each word corresponds to a token) once and performs a "voting task". At the i-th step, it marks (i.e., "votes") with index i each LLT t containing the current (i-th) word of the ADR description. Moreover, it keeps track of the position where the i-th word occurs in t.

MagiCoder tries to find a word match both for the exact and the stemmed version of the meta dictionary and keeps track of the kind of match it has eventually found. It updates a flag, initially set to 0, if at least a stemmed matching is found in an LLT. If a word w has been exactly recognized in a term t, the match between the stemmed versions of w and t is not considered. At the end of the scan, the procedure has built a sub-dictionary containing only terms "voted" at least by one word. We call $\mathsf{Voted}_{\mathtt{LLT}}$ the sub-dictionary of voted terms.

Each voted term t is equipped with two auxiliary data structures, containing, respectively:

1. the positions of the *voting words* in the ADR description; we call voters_t this sequence of indexes;
2. the positions of the *voted words* in the MedDRA term t; we call voted_t this sequence of indexes.

Moreover, we endow each voted term t with a third structure that will contain the *sorting criteria* we define below; we will call it $\mathsf{weights}_t$.

Let us now introduce some notations we will use in the following. We denote as $t.size$ the function that, given an LLT t, returns the number of words contained in t (excluding the stop words). We denote as $\mathsf{voters}_t.length$ (resp. $\mathsf{voted}_t.length$) the function that returns the number of indexes belonging to voters_t (resp. voted_t). We denote as $\mathsf{voters}_t.min$ and $\mathsf{voters}_t.max$ the functions that return the maximum and the minimum indexes in voters_t, respectively.

From now on, sometimes we explicitly list the complete denomination of a terms: we will use the notation *"name"(id)*, where *"name"* is the MedDRA description and *id* is its identifier, that is possibly used to refer to the term. Let us exemplify these notions by introducing an example. Consider the following ADR description: "anaphylactic shock (hypotension + cutaneous rash) 1 hour after taking the drug". Words in it are numbered from 0 (anaphylactic) to 9 (drug). The complete set of data structures coming from the task is too big to be reported here, thus we focus only on two LLTs. At the end of the voting task, $\mathsf{Voted}_{\mathtt{LLT}}$ will include, among others, "Anaphylactic shock" (10002199) and "Anaphylactic reaction to drug" (10054844). We will have that $\mathsf{voters}_{10002199} = [0, 1]$ (i.e., "anaphylactic" and "shock") while $\mathsf{voters}_{10054844} = [0, 9]$ (i.e., "anaphylactic" and "drug"). On the other hand, $\mathsf{voted}_{10002199} = [0, 1]$, revealing that both words in the term have been voted, while $\mathsf{voted}_{10054844} = [0, 2]$, suggesting that only two out of three words in the term have been voted (in particular, "reaction" has not been voted). In this

example all words in the description have been voted without using the stemming.

3.1.4. Weight calculation

After the voting task, selected terms have to be ordered. Notice that a purely syntactical recognition of words in LLTs potentially generates a large number of voted terms. For example, in the Italian version of MedDRA, the word "male" (in English, "pain") occurs 3385 times.

So we have to: i) filter a subset of highly feasible solutions, by means of quantitative weights we assign to candidate solutions; ii) choose a good final selection strategy in order to release a small set of final "winning" MedDRA terms (this latter point will be discussed in Section 3.1.5).

For this purpose, we define four criteria to assign "weights" to voted terms accordingly.

In the following, $\frac{1}{t.size}$ is a normalization factor (w.r.t. the length, in terms of words, of the LLT t). The first three criteria have 0 as optimum value and 1 as worst value, while the fourth criterion has optimum value to 1 and it grows in worse cases.

Criterion one: Coverage

First, we consider how much part of the words of each voted LLT have not been recognized.

$$\mathsf{C}_1(t) = \frac{t.size - \mathsf{voted}_t.length}{t.size}$$

In the example we introduced before, we have that $\mathsf{C}_1(10002199) = 0$ (i.e., all words of the terms have been recognized in the description) while $\mathsf{C}_1(10054844) = 0.33$ (i.e., one word out of three has not been recognized in the description).

Criterion two: Type of Coverage

The algorithm considers whether a perfect matching has been performed with or without using stemmed words. $\mathsf{C}_2(\cdot)$ is simply a flag. $\mathsf{C}_2(t)$ holds if stemming has been used at least once in the voting procedure of t, and it is valued 1, otherwise it is valued 0.

For example, $\mathsf{C}_2(10002199) = 0$ and $\mathsf{C}_2(10054844) = 0$.

Criterion three: Coverage Distance

The use of stemming allows one to find a number of (otherwise lost) matches. As side effect, we often obtain a quite large set of joint winner candidate terms. In this phase, we introduce a string distance comparison between recognized words in the original text and voted LLTs. Among the possible string metrics, we use the so called pair distance [38], which is robust with respect to word permutation. Thus,

$$\mathsf{C}_3(t) = pair(t, \bar{t})$$

where $pair(s, r)$ is the pair distance function (between strings s and r) and $\bar{t}$ is the term "rebuilt" from the words in ADR description corresponding to indexes in voters_t.

For example, $\mathsf{C}_3(10002199) = 0$ (i.e., the concatenation of the voters and the term are equal) and $\mathsf{C}_3(10054844) = 12$.

Criterion four: Coverage Density

We want to estimate how an `LLT` has been covered.

$$\mathsf{C}_4(t) = \frac{(\mathsf{voters}_t.max - \mathsf{voters}_t.min) + 1}{\mathsf{voted}_t.length}$$

The intuitive meaning of the criterion is to quantify the "quality" of the coverage. If an `LLT` has been covered by nearby words, it will be considered a good candidate for the solution. This criterion has to be carefully implemented, taking into account possible duplicated voted words.

After computing (and storing) the weights related to the above criteria, for each voted term t we have the data structure $\mathsf{weights}_t = [\mathsf{C}_1(t), \mathsf{C}_2(t), \mathsf{C}_3(t), \mathsf{C}_4(t)]$, containing the weights corresponding to the four criteria. These weights will be used, after a first heuristic selection, to sort a subset of the syntactically retrieved terms.

Continuing with the example introduced before, we have that $\mathsf{C}_4(10002199) = 1$ while $\mathsf{C}_4(10054844) = 5$. Thus, concluding, we obtain that $\mathsf{weights}_{10002199} = [0, 0, 0, 1]$ while $\mathsf{weights}_{10054844} = [0.33, 0, 12, 5]$.

3.1.5. Selection, ordering and release of winning terms

In order to provide an effective support to pharmacovigilance experts' work, it is important to offer only a small set of good candidate solutions.

As previously said, the pure syntactical recognition of `MedDRA` terms into a free text generates a possibly large set of results. Therefore, the releasing strategy has to be carefully designed in order to select only the best suitable solutions. We will provide a heuristic selection, followed by a sorting of the survived voted terms. Then we propose a release phase of solutions, further refined by a final heuristic criterion.

As a first step, we provide an initial pruning of the syntactically retrieved terms guided by the *ordered-phrases* heuristic criterion. In the ordered-phrases criterion we reintroduce the order of words in the narrative description as a selection discriminating factor. From the set of selected `LLT`s, we remove those terms where voters (i.e., tokens in the original free text) appear in the ADR description in a relative order different from that of the corresponding voted

tokens in the LLT. We do that *only* for those LLTs having voters that voted for more than one term.

Let us consider the following example. On the (Italian) narrative description "edema della glottide-lingua, parestesia al volto, dispnea" (in English, "edema of glottis-tongue, facial paresthesia, dyspnoea"), the voting procedure of MagiCoder finds, among the solutions, the MedDRA terms "Edema della glottide" ("Edema glottis"), "Edema della lingua" ("Edema tongue"), "Edema del volto" ("Edema face"), "Parestesia della lingua" ("Paresthesia tongue"), and "Dispnea" ("Dyspnoea"). The ordererd-phrase criterion removes LLT "Parestesia della lingua" from the set of candidate solutions because "lingua" votes for two terms but in the narrative text it appears before than "parestesia" while in the LLT it appears after.

We call $\mathsf{SelVoted}_{\mathtt{LLT}}$ the set of voted terms after the selection by the ordered-phrases criterion. We proceed then by ordering $\mathsf{SelVoted}_{\mathtt{LLT}}$. We use a sorting on elements $\mathsf{weights}_t$, for each $t \in \mathsf{SelVoted}_{\mathtt{LLT}}$. The obtained subdictionary is dubbed as $\mathsf{SortedVoted}_{\mathtt{LLT}}$ and it has possibly the most suitable solutions on top.

After this phase, the selection of the "winning terms" takes place. The main idea is to select and return a subset of voted terms which "covers" the ADR description. We create the set $\mathsf{Selected}_{\mathtt{LLT}}$ as follows. We iterate on the ordered dictionary and for each $t \in \mathsf{SortedVoted}_{\mathtt{LLT}}$ we select t if all the following conditions hold:

1. t is completely covered, i.e., $\mathsf{C}_1(t) = 0$;
2. t does not already belong to $\mathsf{Selected}_{\mathtt{LLT}}$;
3. t is not a prefix of another selected term $t' \in \mathsf{SortedVoted}_{\mathtt{LLT}}$;
4. t has been voted without stemming (i.e., $\mathsf{C}_2(t) = 0$) or, for any $w_i \in \mathsf{voters}_t$, w_i has not been covered (i.e., no term voted by w_i has been already selected) or w_i has not been exactly covered (i.e., only its stem has been recognized in some term t_1)[9].

At this stage, we have a set of MedDRA terms which "covers" the narrative description. We further select a subset $\mathsf{FinalVoted}_{\mathtt{LLT}}$ of $\mathsf{Selected}_{\mathtt{LLT}}$ with a second heuristic, the *maximal-set-of-voters* criterion.

The maximal-set-of-voters criterion deletes from the solution those terms which can be considered "extensions" of other ones. For each pair of terms t_i and t_j, it checks if voters_{t_i} is a subset of voters_{t_j} (considered as sets of indexes). If it is the case, t_i is removed from $\mathsf{Selected}_{\mathtt{LLT}}$.

[9] In the implementation we add also the following thresholds: we choose only terms t such that $\mathsf{C}_3(t) < 0.5$ and $\mathsf{C}_4(t) < 3$. We extracted these thresholds by means of some empirical tests. We plan to eventually adjust them after some further performance tests.

In MagiCoder we do not need to consider ad-hoc subroutines to address permutations and combinations of words (as it is done, for example, in [20]). In Natural Language Processing, permutations and combinations of words are important, since in spoken language the order of words can change w.r.t. the formal structure of sentences. Moreover, some words can be omitted, while the sentence still retains the same meaning. These aspects come for free from our voting procedure: after the scan, we retrieve the information that *a set of words covers a term* $t \in \mathsf{Voted}_{\mathtt{LLT}}$, but the order between words does not necessarily matter.

3.2. MagiCoder: structure of the algorithm and Pseudocode

Figure 2 depicts the pseudocode of MagiCoder. We represent dictionaries either as sets of words or as sets of functions. We describe the main procedures and functions used in the pseudocode.

- Procedure *Preprocessing* takes the narrative description, performs tokenization and stop-word removal and puts it into an array of words.
- Procedures *CreateMetaDict* and *CreateMetaDictStem* get LLTs and create a dictionary of *words* and of their stemmed versions, respectively, which belong to LLTs, retaining the information about the set of terms containing each word.
- By the functional notation $\mathsf{DictByWord}(w)$ (resp., $\mathsf{DictByWordStem}(w)$), we refer to the set of LLTs containing word w (resp., the stem of w).
- Function $stem(w)$ returns the stemmed version of word w.
- Function $indx_t(w)$ returns the position of word w in term t.
- $stem\_usage_t$ is a flag, initially set to 0, which is set to 1 if at least a stemmed matching with the MedDRA term t is found.
- $\mathsf{adr\_clear}$, voters_t, voted_t are arrays and $\mathsf{add}[A, l]$ appends l to array A, where l may be an element or a sequence of elements.
- C_i $(i = 1, 2, 3, 4)$ are the weights related to the criteria defined in Section 3.1.4.
- Procedure $sortby(A, \{v_1, \ldots, v_k\})$ performs the sorting of the array A based on the values of properties $v_1, \ldots, v_k$ of its elements.
- Procedure $prefix(S, t)$, where S is a set of terms and t is a term, tests whether t (considered as a string) is *prefix* of a term in S. Dually, procedure $remove\_prefix(S, t)$ tests whether in S there are one or more prefixes of t, and eventually removes them from S.

- Function $mark(w)$ specifies whether a word w has been already covered (i.e., a term voted by w has been selected) in the (partial) solution during the term release: $mark(w)$ returns 1 if w has been covered (with or without stemming) and it returns 0 otherwise. We assume that before starting the final phase of building the solution (i.e., the returned set of `LLT`s), $mark(w) = 0$ for any word w belonging to the description.

- Procedures $ordered\_phrases(S)$ and $maximal\_voters(S)$, where S is a set of terms, implement *ordered-phrases* and *maximal-set-of-voters* criteria (defined in Section 3.1.5), respectively.

- Function $win(S, n)$ returns the first n elements of an ordered set S. If $|S| = m < n$, the function returns the complete list of ordered terms and $n - m$ nil values.

3.3. MagiCoder complexity analysis

Let us now conclude this section by sketching the analysis of the computational complexity of MagiCoder.

Let n be the input size (the length, in terms of words, of the narrative description). Let m be the cardinality of the dictionary (i.e., the number of terms). Moreover, let m' be the number of distinct words occurring in the dictionary and let k be the length of the longest term in the dictionary. For `MedDRA`, we have about 75K terms (m) and 17K unique words (m'). Notice that, reasonably, k is a small constant for any dictionary; in particular, for `MedDRA` we have $k = 22$. We assume that all update operations on auxiliary data structures require constant time $O(1)$.

Building meta-dictionaries DictByWord and DictByWordstems requires $O(km)$ time units. In fact, the simplest procedure to build these hash tables is to scan the `LLT` dictionary and, for each term t, to verify for each word w belonging to t whether w is a key in the hash table (this can be done in constant time). If w is a key, then we have to update the values associated to w, i.e., we add t to the set of terms containing w. Otherwise, we add the new key w and the associated term t to the hash table. We note that these meta-dictionaries are computed only once when the `MedDRA` dictionary changes (twice per year). Then as many narrative texts as we want can be encoded without the need to rebuild these meta-dictionaries.

Hash tables permit a fast lookup for words and stems, drastically reducing computational time complexity. On the other hand, they require memory space. However we note that in our tests using `MedDRA`, the memory usage has not been excessive (about 200MB) with respect to memory usually available in current PCs and servers. Moreover, we note that these hash tables are mainly used for meta-dictionaries, thus they may be shared across several instances of MagiCoder responding to different encoding requests.

It can be easily verified that the voting procedure requires in the worst case $O(nm)$ time units. This is a totally conservative bound, since this worst case should imply that each word of the description appears in all the terms of the

```
Procedure MagiCoder(D text, LLTDict dictionary, n integer)
Input: D: the narrative description;
       LLTDict: a data structure containing the MedDRA LLTs;
       n: the maximum number of winning terms that have to be released by the procedure
Output: an ordered set of LLTs
DictByWord = CreateMetaDict(LLTDict);
DictByWordStem = CreateStemMetaDict(LLTDict);
adr_clear = Preprocessing(D);
adr_length = adr_clear.length;
Voted_LLT = ∅;
/* for each non-stop-word in the description                                   */
foreach (i ∈ [0, adr_length − 1] do
    /* test whether the current word belongs to MedDRA                        */
    if adr_clear[i] ∈ DictByWord then
        /* for each term containing the word                                  */
        foreach t ∈ DictByWord(adr_clear[i]) do
            /* keep track of the index of the voting word                     */
            add[voters_t,i];
            /* keep track of the index of the recognized word in t            */
            add[voted_t, indx_t(adr_clear[i])];
            Voted_LLT = Voted_LLT ∪ t;
    /* test if the current (stemmed) word belongs the stemmed MedDRA          */
    if stem(adr_clear[i]) ∈ DictByWordStem then
        foreach t ∈ DictByWordStem(stem(adr_clear[i])) do
            /* test if the current term has not been exactly voted by the same word */
            if i ∉ voters_t then
                add[voters_t, i];
                add[voted_t, indx_t(adr_clear[i])];
                /* keep track that t has been covered by a stemmed word       */
                stem_usage_t = true;
            Voted_LLT = Voted_LLT ∪ t
/* for each voted term, calculate the four weights of the corresponding criteria */
foreach t ∈ Voted_LLT do
    add[weights_t, C1(t), C2(t), C3(t), C4(t)]
/* filtering of the voted terms by the first heuristic criterion              */
SelVoted_LLT = orderd_phrases(Voted_LLT);
/* multiple value sorting of the voted terms                                  */
SortedVoted_LLT = sortby(SelVoted_LLT, {C1, C2, C3, C4});
foreach t ∈ SortedVoted_LLT do
    foreach index ∈ voters_t do
        /* select a term t if it has been completely covered, its i-th voting word has not been covered
           or if its i-th voting word has been perfectly recognized in t and if t is not prefix of another
           already selected terms                                             */
        if C1(t) = 0 AND ((stem_usage_t = false OR (mark(adr_clear(index))=0)) AND
           t ∉ Selected_LLT AND prefix(Selected_LLT,t)=false) then
            mark(adr_clear(index))=1;
            /* remove from the selected term set all terms which are prefix of t */
            Selected_LLT = remove_prefix(Selected_LLT,t);
            Selected_LLT = Selected_LLT ∪ t
/* filtering of the finally selected terms by the second heuristic criterion  */
FinalVoted_LLT = maximal_voters(Selected_LLT);
winners = win(FinalVoted_LLT, n);
return winners
```

Figure 2: Pseudocode of MagiCoder

dictionary. A simple analysis of the occurrences of the words in MedDRA shows that this worst case never occurs. In fact, the maximal absolute frequency of a MedDRA word is 3937, and the average of the frequencies of the words is 19.1

(These values have been calculated excluding the stop-words and taking into account the stems of words appearing in MedDRA). Thus, usually, real computational complexity is much less of this worst case.

The computation of criteria-related weights requires $O(nm)$ time units. In particular, both criterion one and criterion two require $O(m)$ time units. Criterion three requires $O(nm)$ units (we assume to absorb the complexity of the pair distance function); criterion four requires $O(nm)$ time units.

The subsequent sorting based on computed weights is a sorting algorithm having complexity approximated to $O(m \log m)$, based on the comparison of objects of four elements (i.e., the weights of the four criteria). Since the number of the criteria-related weights involved in the sorting is constant, it can be neglected. Thus, the complexity of sorting can be considered to be $O(m \log m)$.

Finally, to derive the best solutions actually requires $O(nm)$ steps. The ordered-phrases criterion requires $O(nm)$; the maximal set of voters criterion takes $O(mn)$ time units.

Thus, we conclude that MagiCoder requires in the worst case $O(nm)$ computational steps. We again highlight that this is a (very) worst-case scenario, while in average it performs quite better. Indeed, we did not take into account that each phase works on a subset of terms of the previous phase, and the size of these subsets rapidly decreases in real computations. The selection phase works only on voted terms, thus, in common applications, on a subset of the original dictionary.

3.4. Software implementation: the user interface

MagiCoder has been implemented as a VigiFarmaco plug-in: people responsible for pharmacovigilance can consider the results of the auto-encoding of the narrative description and then revise and validate it. Figure 3 shows a screenshot of VigiFarmaco during this task. In the top part of the screen it is possible to observe the five sections composing a report. The screenshot actually shows the result of a human-MagiCoder interaction: by pressing the button "Autocodifica in MedDRA" (in English, "MedDRA auto-encoding"), the user responsible for pharmacovigilance obtains a MedDRA encoding corresponding to the natural language ADR in the field "Descrizione" (in English, "Description"). Up to six solutions are proposed as the best MedDRA term candidates returned by MagiCoder. According to the processing description, MagiCoder can also return an empty solution (if no ADRs are detected). If the automatic encodings reveals more than six MedDRA terms, the sofware returns the first six in the ordered list of candidate terms (see Section 3.1). The user can refuse a term (through the trash icon), change one or more terms (by an option menu), or simply validate the automatic encoding and switch to the next section "Farmaci" (in English, "Drugs"). The maximum number of six terms to be shown has been chosen in order to supply pharmacovigilance experts with a set of terms extended enough to represent the described adverse drug reaction but not so large to be redundant or excessive.

We are testing MagiCoder performances in the daily pharmacovigilance activities. Preliminary qualitative tests and discussions with people responsible for

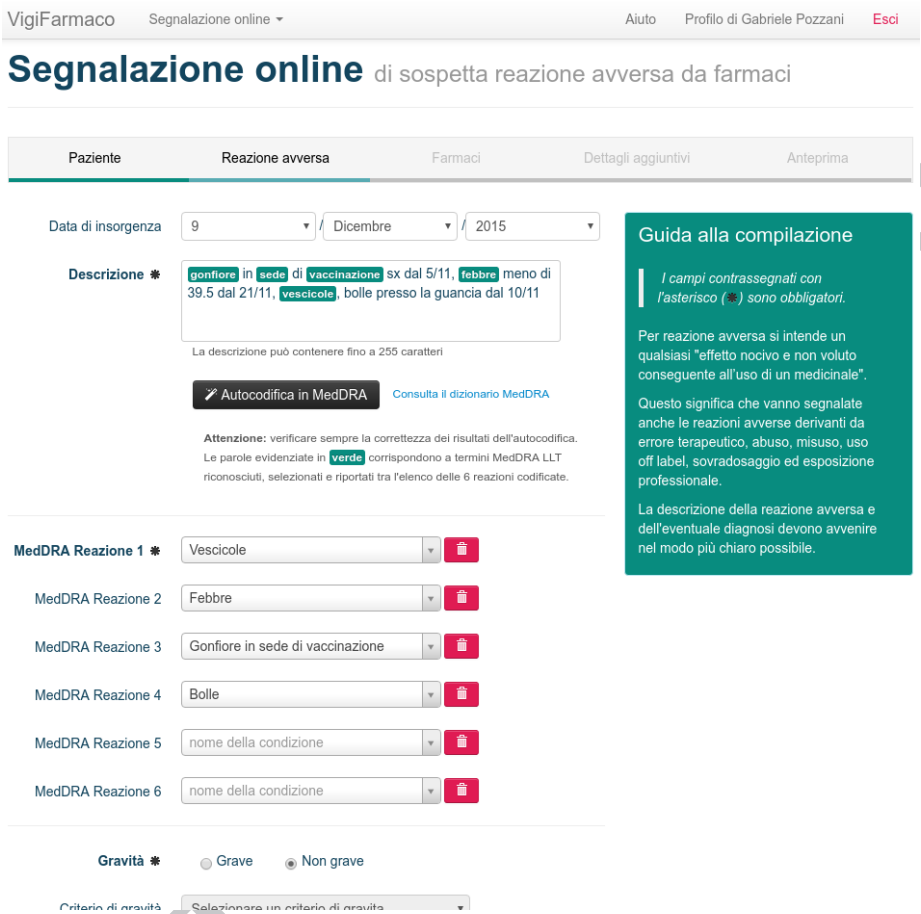


Figure 3: A partial screenshot of `VigiFarmaco` User Interface. The field *Descrizione* (*Description*) is devoted to the natural language description. Below it can noticed the automatic encoding into four `MedDRA` terms.

pharmacovigilance who are using VigiFarmaco show that MagiCoder drastically reduces the amount of work required for the revision of a report, allowing the pharmacovigilance stakeholders to provide high quality data about suspected ADRs.

4. Testing MagiCoder performances

In this section we describe the experiments we performed to evaluate MagiCoder performances. The first completely automatic test exploits a large amount of manually revised reports we obtained from VigiSegn [39] (see Section 4.1). The second test involves a set of about 1800 reports, revised ex novo by a group of three experts of the domain, that represents a gold standard and provides

Precision	$\mathtt{P} = \frac{\lvert \mathsf{RelS} \cap \mathsf{RetS} \rvert}{\lvert \mathsf{RetS} \rvert} = \frac{TP}{TP+FP}$
Recall	$\mathtt{R} = \frac{\lvert \mathsf{RelS} \cap \mathsf{RetS} \rvert}{\lvert \mathsf{RelS} \rvert} = \frac{TP}{TP+FN}$

Table 2: Performance and correctness measures

a sound reference for the accurate estimation and classification of MagiCoder errors.

In information retrieval the standard approach to system evaluation revolves around the orthogonal notions of relevant/non-relevant solution and retrieved/non-retrieved solution [40]. In our setting, a solution is *relevant* if and only if it is a `MedDRA` term which correctly encode the narrative description. On the other hand, a *retrieved solution* is trivially defined as a MagiCoder output term, independently from its relevance. We dub the sets of relevant and retrieved solutions as RelS and RetS, respectively. Two main kinds of errors are defined [40]: *false positive errors* (FP) and *false negative errors* (FN). In our setting, these errors can be defined as follows. A false positive error corresponds to the retrieval of a "wrong" `LLT`, i.e., a retrieved non-relevant term. A false negative error corresponds to the failure in the recognition of a "good" `LLT`, i.e., a non-retrieved relevant term.

For example, consider a report that has the (partially misspelled) narrative form "Seizure followed by headace" (notice the typo in the last word). It may be mapped to two MedDRA terms, say, "Seizure" (10039906) and "Headache" (10019211). MagiCoder fails in recognise the misspelled word "headace" as the correct corresponding `LLT` and only returns "Seizure". Then, in the MagiCoder's output we measure (w.r.t. a correct encoding) a true positive ("Seizure") and a false negative ("Headache").

As dual notions of false positive and false negative errors, one can define *correct* results, i.e., *true positive* (TP) and *true negative* (TN): in our case, a true positive corresponds to a correctly returned relevant `LLT`, and a true negative is an non-relevant `LLT`, which correctly has not been recognized as a solution.

The evaluation of the false positive and the false negative rates, and in particular of the impact of relevant solutions among the whole set of retrieved solutions, are crucial measures in order to estimate the quality of the automatic encoding. For this purpose two main metrics are used, i.e., precision and recall. The *precision* (P), also called positive predictive value, is the percentage of retrieved solutions that are relevant. The *recall* (R), also called sensitivity, is the percentage of all relevant solutions that have been retrieved by the system.

Table 2 summarizes formulas for precision and recall. We provide formulas both in terms of relevant/retrieved solutions and binary classification errors.

It is worth noting that the binary classification of solutions as relevant or non-relevant is referred to as the gold standard judgment of relevance. In our case, the gold standard is represented by a *human encoding* of the narrative description, i.e., a set of `MedDRA` terms choosen by a pharmacovigilance expert.

Class	# chars	# reports	Recall R	Precision P
1	0-20 chars	459	86%	88%
2	21-40 chars	1012	72%	75%
3	41-100 chars	1993	61%	62%
4	101-255 chars	970	58%	52%
5	>255 chars	11	46%	45%
overall		4445	65%	65%

Table 3: Results of the first experiment on MagiCoder

Such a set is assumed to be definitively *correct* (only correct solutions are returned) and *complete* (all correct solutions have been returned).

4.1. First experiment

To evaluate MagiCoder performances, we first developed a benchmark, which automatically compares MagiCoder behavior with human encoding on already manually revised and validated ADR reports.

For this purpose, we exploited VigiSegn, a data warehouse and OLAP system that has been developed for the Italian Pharmacovigilance National Center [39]. VigiSegn offers a large number of *encoded* ADR reports. The encoding has been manually performed and validated by experts working at pharmacovigilance centers. Encoding results have then been sent to the national regulatory authority, AIFA. The dataset we considered covers all the 4445 reports received, revised, and validated during 2014 in the Italian region Veneto.

To test MagiCoder we *automatically* compared the output provided by the procedure with the manual encoding of the ADR reports from the dataset. For each report, we checked which terms belong to either both or only one of manual and automatic solutions, and then we calculated errors and performance measures. In order to have two uniform data sets, we compared only those reports where MagiCoder recognized at most six terms, i.e., the maximum number of terms that human experts are allowed to select through the `VigiFarmaco` user interface.

4.1.1. Results

Table 3 shows the results of this first performance test. We grouped narrative descriptions by increasing length (in terms of characters). Reported results are computed considering terms at `PT` level, i.e., mapping each `LLT` recognized by the human experts or by MagiCoder to its corresponding preferred term. By moving to `PT` level, instead of using the `LLT` level, we group together terms that represent the same medical concept (i.e., the same adverse reaction). In this way, we do not consider an error when MagiCoder and the human expert use two different `LLT`s for representing the same adverse event. The use of the `LLT` level for reporting purpose and the `PT` level for analysis purpose is suggested also by `MedDRA` [3].

MagiCoder behaves very well on very short (class 1) and short (class 2) descriptions. Recall and precision remain greater than 50% up to class 4. Very long descriptions (class 5), on which performances drastically decrease, represent a negligible percentage of the whole set (less than 0.3%).

It is worth noting that this test simply estimates how much, for each report, the MagiCoder behavior is similar to the manual work. The query we perform to compare manual and automatic encoding is, obviously, quantitative. For each VigiSegn report, the query is able to detect common retrieved terms and terms returned either by the human expert or by MagiCoder. It is not able to fairly test *redundancy* errors, human experts often make some encoding choices in order to avoid repetitions. Thus, an LLT returned by MagiCoder that has not been selected by the expert because redundant is not truly a false positive. This suggests that we are probably underestimating MagiCoder performances.

4.1.2. Examples

Table 4 provides some examples of the behavior of MagiCoder. We propose some Italian free text ADR descriptions from the first experiment and provide both the manual and the automatic encodings into LLT terms. To ease the reading, we also provide a quite straightforward *literal* translation in English of reports and encodings.

In Table 4 we use the following notations: $\underline{t_1}^n$ and $\underline{t_2}^n$ are two *identical* LLTs retrieved both by the human and the automatic encoding; $\overline{t_1}^n$ and $\overline{t_2}^n$ are two *semantically equivalent* (i.e., LLTs with the same PT) or *similar* LLTs retrieved by both the human and the automatic encoding and considered as true positive; we use italic to denote text in the descriptions and LLTs that have been recognized only by MagiCoder. For example, in description D3, "cefalea" (in English, "headache") is retrieved and encoded both by the human expert and MagiCoder; in description D2, ADR "febbre" (in English, "fever') has been encoded with the term itself by the algorithm, whereas the expert encoded it with its synonym "piressia" (in English, "pyrexia"); in D1, ADR "ipotensione" (in English, "hypotension") has been retrieved only by MagiCoder.

To exemplify how the ordered-phrase heuristic works, we can notice that in D2 MagiCoder did not retrieve the MedDRA term "Vescicole in sede di vaccinazione" (10069623), Italian for "Vaccination site vesicles". It belongs to the set of the voted solutions (since $\mathsf{C}_1(10069623) = 0$), but it has been pruned from the list of the winning terms by the ordered-phrase heuristic criterion.

4.2. Second experiment

For the second experiment, we created a gold reference standard with the involvement of three experts of the domain. The experts have a great experience both in report annotation and in the use of MagiCoder. We adopted a standard praxis: after the blind revision of two experts, which worked separately, a third moderator reviewed and eventually corrected and made the gold standard uniform.

#	Narrative description	LLT human encoding	LLT MagiCoder encoding
D1	Shock anafilattico (*ipotensione* + rash cutaneo) 1 h dopo assunzione x os del farmaco	Shock anafilattico[1]	*Ipotensione*, Shock anafilattico[1]
	Anaphylactic shock (*hypotension* + rash cutaneous) 1 hour after assumption per os of the medicine	Anaphylactic shock[1]	*Hypotension*, Anaphylactic shock[1]
D2	Gonfiore in sede di vaccinazione sx dal 5/11, febbre meno di 39,5 dal 21/11, vescicole, *bolle* presso la guancia dal 10/11	Piressia[1], Vescicole[2], Gonfiore in sede di vaccinazione[3]	*Bolle*, Febbre[1], Vescicole[2], Gonfiore in sede di vaccinazione[3]
	Swelling in the area of left vaccination from 5/11, fever less than 39.5 since 21/11, blisters, *bullaes* at the cheek since 10/11	Pyrexia[1], Blisters[2], Swelling in vaccination site[3]	*Bullaes*, Fever[1], Blisters[2], Swelling in vaccination site[3]
D3	Reazione locale estesa, *dolore* locale; cefalea e febbre per due giorni	Cefalea[1], Febbre[2], Reazione in sede di vaccinazione[3]	Cefalea[1], *Dolore*, Febbre[2], Reazione locale[3]
	Extended local reaction, local *pain*; headache and fever for two days	Headache[1], Fever[2], Vaccination site reaction[3]	Headache[1], *Pain*, Fever[2], Local reaction[3]

Table 4: Examples of MagiCoder behavior

The dataset includes 1805 ADR narrative descriptions from VigiSegn. We grouped reports by increasing length as in the previous experiment, taking into account different length intervals. The definition of the classes and the number of reports in each of them are reported in Table 5.

The cardinality of classes does not reflect the effective distribution of lengths among the whole dataset. In particular, very long ADR description (>199 characters) represents a small percentage. Notwithstanding, we decided to observe MagiCoder behaviour on a significative amount of long (and potentially tricky) documents.

Pharmacologists reviewed the reports ex novo and then compared results with MagiCoder outputs. In the comparison, they considered redundancy and semantic equivalence of different solutions as right encoding, and so provided a fair account of MagiCoder performances.

4.2.1. Results

Results of the second experiment are summarized in Table 5.

Moreover, they taxonomized false positives and false negatives errors into more refined subclasses. Among false negatives, they distinguished the following three cases.

FNi) *Laboratory results.* Out of range results for a lab test described in the

Class	# chars	# reports	Recall R	Precision P
1	<50 chars	405	93.6%	98.7%
2	50-100 chars	368	87.0%	96.6%
3	100-149 chars	378	87.9%	92.5%
4	150-199 chars	358	87.8%	89.7%
5	>199 chars	301	84.4%	86.8%
overall		1810	87.7%	92.0%

Table 5: MagiCoder performances according to the second experiment

ADR report may entail some adverse reaction (e.g., systolic blood pressure greater than 140 stands for systolic hypertension), that cannot yet be interpreted by MagiCoder.

FNii) *Mispelled words.* Since MagiCoder uses perfect string matching, it is not still able to recognize mispelled words and then it may miss an adverse reaction.

FNiii) *Lack of synonyms.* Missing terms are due to the lack of synonymical terminology.

Similarly, also false positive are classified into three classes.

FPi) *Therapeutic indication/concomitant pathologies.* Sometimes ADR reports contain also information about either concomitant pathologies or the therapeutic indication of the administered drugs. These clinical data may be wrongly interpreted by MagiCoder as ADRs.

FPii) *Negation.* MagiCoder does not take negations into account, thus a reaction may be reported when its absence is stated in the ADR report.

FPiii) *Similar words.* Some errors are induced by the usage of syntactically similar words with different meanings that MagiCoder is not able to discriminate.

Some examples of these error subclasses can be found in Tables 6, 11, and 12. Since reviewers can evaluate the semantic equivalence of apparently different results, human comparison is certainly more fair than the automatic one performed in the first experiment. For this reason, with respect to performances calculated in the first experiment, MagiCoder performances degrade less moving from class 1 to class 5, and the global results are drastically better. Indeed, the recall and the precision reach 86.99% and 91.81%, respectively.

Analyzing errors, we observe that among false negatives FNi, FNii, and FNiii errors represent the 3%, 8%, and 89%, respectively. This confirms the importance to equip MagiCoder with synonyms (see Section 5.2). Among false positives, the majority of errors is covered by FPi (61%) whereas FPii and FPiii represent 16% and 23% of errors, respectively.

Errors of type FPi (i.e., errors involving therapeutic indication/concomitant pathologies) can only be addressed by extending MagiCoder with suitable knowledge bases. Since therapeutic indications are systematically included in the description of ADRs, the same kind of errors can be observed also on the dataset of English Summaries of Product Characteristics analyzed in Section 4.3.

Errors related to the lack of the negation detection (FPii) represent a smaller percentage. A discussion about the detection of connectives is reported in Section 5.3.

Finally, errors involving similar words depend on the given language and partially on the stemmer algorithm. It seems quite tricky to find a general solution without a drastic worsening of MagiCoder performances. A partial, but potentially satisfactory solution, may be to handle with ad-hoc solutions frequently occurring cases (as the common Italian false positive *mania* described in Section 3.1.2).

4.2.2. Examples

Table 6 reports three examples of ADR descriptions from the second experiment. Each `LLT` is eventually annotated with its error class as reported by domain experts.

Apart errors, we note in the first ADR how experts may consider correct MagiCoder outputs also when they do not match perfectly the gold standard. In the ADR description, the reporter said "ostruzione delle coane da secrezione" (in English, "choanae obstruction due to secretion"). Experts encoded that part of the ADR as "Ostruzione nasale" (Italian for "Nasal obstruction") but accepted as correct the MagiCoder encoding "Ostruzione" (in English, "Obstruction") although the anatomical descriptor ("nasale") is missing.

Similarly, in the third ADR, we note that experts did not considered incorrect the output of MagiCoder, despite it is redundant. Indeed, all encoded reactions are outcomes of "Shock anafilattico" (in English, "Anaphylactic shock") and thus they are omitted in the gold standard.

4.3. Some initial experiments on English text

MagiCoder has been developed to be robust with respect to language changes. Indeed, no assumptions related to the structure of written phrases are directly exploited in the procedure.

The full exploration of MagiCoder performances and its potentiality on English language is one of the main current developments of MagiCoder. So far, we obtained some evidences about the good behavior of the software also for English written descriptions. In order to test MagiCoder, one needs a large corpus (i.e., *annotated English text*), possibly a set of narrative descriptions about suspected ADRs.

Up to now, we performed two preliminary experiments. They suggest promising results also on English written clinical texts. In both experiments, no synonyms for English `MedDRA` have been considered.

ADR description	Gold standard	MagiCoder
Dopo circa 5/6' dalla fine della somministrazione: dispnea, sensazione di soffocamento, ostruzione delle coane da secrezione, eruzione cutanea al volto. No broncospasmo	Sensazione di soffocamento, Dispnea, Ostruzione nasale, Eruzione cutanea	Secrezione (FPi), Broncospasmo (FPii), Dispnea, Ostruzione, Eruzione cutanea, Sensazione di soffocamento
Eritema cutaneo, atralgie migranti, rialzo degli indici di funzione renale (rialzo della creatinina di 1mg/dl), rialzo di VES e PCR	Velocità di eritrocitosedimentazione aumentata (FNiii), Creatinina aumentata (FNiii), Proteina C aumentata (FNiii), Artralgia (FNii), Eritema della cute (FNiii)	Eritema migrante (FPiii)
Il paziente riferisce reazione allergica, vomito, diarrea, orticaria gigante e inizio di shock anafilattico, bolle pruriginose	Shock anafilattico	Bolle, Vomito, Diarrea, Reazione allergica, Orticaria gigante, Shock anafilattico

Table 6: Examples of Adverse Drug Reaction descriptions from the second experiment

The first experiment involves some examples proposed in [20], where TextMiner, a text extraction tool for MedDRA and WHO-ART[10] terminologies, is described and implemented. In [20], the author addressed the same problem we are facing, i.e., the encoding of free text descriptions of adverse drug reactions. The comparison has been done on a small data set, and is based on some Martindale [23] narrative sentences, the author considered in [20]. We considered only sentences effectively describing adverse effects.

In Tables 7, 8, and 9 we compare the behavior of MagiCoder and TextMiner on the same input and we can observe that MagiCoder exceeds TextMiner performances. Recall that TP stands for True Positives, FP stands for False Positives and FN stands for False Negatives. Let us now discuss interesting differences, strong and weak points of MagiCoder encoding.

Martindale description 1. CNS-related adverse effects include headache, vertigo, dizziness, nervousness, tinnitus, depression, drowsiness, and insomnia. Hypersensitivity reactions may occur occasionally and include fever, angioedema, bronchospasm, and rashes. Hepatotoxicity and aseptic meningitis, which occur rarely, may also be hypersensitivity reactions. Some patients may experience visual disturbances.

[10] https://www.nlm.nih.gov/research/umls/sourcereleasedocs/current/WHO/

	MagiCoder	TextMiner
TP	Headache, Vertigo, Dizziness, Insomnia, Nervous, Tinnitus, Depressive reaction, Angioedema, Bronchospasm, Fever, Rash all over, Aseptic meningitis, Drowsiness, Hepatotoxicity, Hypersensitivity reaction, Visual disturbances	Headache, Vertigo, Dizziness, Insomnia, Nervousness, Tinnitus, Depression, Angioedema, Bronchospasm, Fever, Rash, Meningitis
FP	No adverse effect	Meningism
FN		Drowsiness, Hypersensitivity reaction, Hepatotoxicity, Visual disturbances

Table 7: Comparison of MagiCoder and TextMiner on Martindale description 1

The MagiCoder false positive "No adverse effect" is, de facto, an uninformative (in our setting) MedDRA term. A manual cleaning of uninformative preferred terms (and related LLTs) can easily avoid this kind of error.

Martindale description 2. Reports have included anaphylaxis (which has sometimes been fatal, and may occur after the first dose), serum sickness, Stevens-Johnson syndrome, toxic epidermal necrolysis (sometimes fatal), laryngeal oedema, and vasculitis.

	MagiCoder	TextMiner
TP	Vasculitis, Laryngeal oedema, Serum sickness, Stevens-Johnson syndrome, Toxic epidermal necrolysis, Anaphylaxis	Vasculitis, Oedema, Serum sickness, Stevens Johnson syndrome, Epidermal necrolysis
FP		Laryngitis
FN		Anaphylaxis

Table 8: Comparison of MagiCoder and TextMiner on Martindale description 2

Martindale description 3. Severe life-threatening events, including hyperosmolar nonketotic hyperglycaemic coma, diabetic ketoacidosis, hypoglycaemic coma, convulsions, and mental status changes have been reported very rarely

	MagiCoder	TextMiner
TP	Hypoglycaemic coma, Convulsions, Mental status changes, Diabetic ketoacidosis	Hypoglycaemic coma, Convulsions
FP	Diabetic coma, Coma	Coma diabetic
FN	Hyperglycaemic coma	Hyperglycaemic coma, Diabetic ketoacidosis, Mental status changes

Table 9: Comparison of MagiCoder and TextMiner on Martindale description 3

Product	Active ingredients	Recall R	Precision P
000546	Pregabalin	97.67%	92.31%
001026	Liraglutide	92.50%	82.22%
002048	Collagenase clostridium histolyticum	88.43%	85.96%
002332	Boceprevir	99.21%	94.38%
002601	Dimethyl fumarate	88.46%	60.53%
002783	Levetiracetam	90.79%	81.18%
003766	Evolocumab	81.82%	56.25%
003768	Daclatasvir dihydrochloride	94.60%	92.10%
003985	Nivolumab	87.94%	84.93%
004042	Elvitegravir / Cobicistat / Emtricitabine / Tenofovir alafenamide	62.07%	72.00%
Average		88.35%	80.20%

Table 10: MagiCoder performances on a dataset of ten English SPCs

The second experiment involves a set of ten Summaries of Product Characteristics (SPCs) from the European Medicines Agency[11] (EMA), manually revised by an expert of the domain.

Table 10 summarizes MagiCoder performances on this dataset.

We analyze here one of the worst performances, measured on product 004042, where the recall reaches only 62%. We focus on some examples of false negative errors and we discuss the MagiCoder behavior case by case.

We categorized errors according to the classification defined in Section 4.2 Moreover, we add two further classes of errors:

FNiv) false negative errors related to MagiCoder selection criteria and heuristics.

FPiv) false positive errors related to MagiCoder selection criteria and heuristics.

Some false negatives and false positives on SPC 004042 are reported in Tables 11 and 12, respectively. For each error we propose the original text fragment from the SPC, the expert's encoding and, eventually, a comment about the error made by MagiCoder.

[11] http://www.ema.europa.eu/

FN MedDRA term	**Type of error**	**Original text in the SPC**	**Comment**
Anaemia (10002034)	FNii	anaemia1	Mispelled word, not recognized by the perfect matching
Cholesterol total increased (10008671)	FNiv	The median (Q1, Q3) change from baseline in total cholesterol to HDL-cholesterol ratio at Week 96 was 0.1 (-0.3, 0.7)	a word ("increased") of the LLT 10008671 has not been voted, then the selection rejects the LLT (treshold on criterion $C_1(\cdot)$)
Serum creatinine abnormal (10040231)	FNiii	Changes in serum creatinine	semantic information/ synonym needed
Infection reactivation(10070891)	FNiii	Immune Reactivation Syndrome	semantic information/ synonym needed

Table 11: False Negative Errors in SPC 004042

FP MedDRA term	**Type of error**	**Original text in the SPC**	**Comment**
HIV infection (10020161)	FPi	In HIV infected patients with severe immune deficiency at the time of initiation ..	The description of the potential concomitant cronic condition is encoded by MagiCoder
Rash all over (10037848)	FPiv	rash	*all* and *over* are both considered as stop words and the term is considered as completely recognised and selected
Renal impairment (10062237)	FPi	The safety profile of Genvoya in patients with mild to moderate renal impairment was similar to that in patients with normal renal function	Similar to case (10020161)

Table 12: False Positive Errors in SPC 004042

5. Discussion

We discuss here some choices and issues we faced when developing MagiCoder.

5.1. Stemming and performance of the NLP software

Stemming is a useful tool for natural language processing and text searching and classification. Deriving the stemmed form of a word is a non-trivial operation. In particular, due to the complexity of the language and the number of linguistic variations and exceptions, stemming for Italian language is extremely critical.

For the first implementation of MagiCoder as `VigiFarmaco` plug-in, we used a robust implementation of the Italian stemming procedure[12] based on Porter's algorithm [21]. The procedure takes into account subtle properties of the language and the etymological source of the words. In addition to the simple recognition of words up to plurals and genres, it is able, in most cases, to recognize an adjectival form of a noun by extracting the same syntactical root.

Despite the efficiency of this algorithm, we noticed that the recognition of some `MedDRA` terms were lost. In some sense, this stemming algorithm is too "aggressive" and, in some cases, counterintuitive. For example, the Italian adjective "psichiatrico" (in English, "psichiatric") and its plural form "psichiatrici" have two different stems, "psichiatr" and "psichiatric", respectively. Thus, in this case the stemmer fails in recognizing the singular and plural forms of the same word.

We then decided to adopt the stemming algorithm also used in Apache Lucene[13], an open source text search engine library. This procedure is less sophisticated w.r.t. the stemming algorithm cited above, and can be considered as a "light" stemmer. It simply elides the final vowels of a word [41]. This induces a conservative approach and a uniform processing of the whole set of `MedDRA` words. This may be unsatisfactory for a general problem of text processing, but it is fruitful in our setting. We repeated the first experiment (the completely automatic one on the VigiSegn dataset) both with the classical and the light stemmer. In the latter case, we measured a global enhancement of the performances. Average recall and precision on all reports (i.e., without regards to their length), moved from 63% and 61%, respectively, to 65% (overall performances in Table 3). We note in particular the growth of recall and precision on class 5, where an increase of 5% and 7% have been measured, respectively.

5.2. Synonyms

MagiCoder performs a pure syntactical recognition (up to stemming) of the words in the narrative description. No semantic information is used in the current version of the algorithm. On the other hand, in written informal language,

[12] http://snowball.tartarus.org/
[13] https://lucene.apache.org/

synonyms and variations of words and locutions (e.g., acronyms, abbreviations) are frequently used. Thus, a natural evolution of our NLP software may be the addition of an Italian thesaurus dictionary. This would appear a trivial extension, as one could try to match MedDRA both with original words and their synonyms, and try to maximize the set of retrieved terms. We performed a preliminary test and we observed a drastic deterioration of performances (both in terms of correctness and completeness). The main reason is related to the nature of natural language. Indeed, synonymical groups include words related by figurative meaning. For example, among the synonyms of word "faccia" (in English, "face"), one finds "viso" (in English "visage"), which is semantically related, but also "espressione" (in English, "expression"), which is not relevant in the considered medical context. Moreover, the use of synonyms of words in ADR text leads to an uncontrolled growth of the voted terms, that barely can be later dropped in the final terms release. Furthermore, the word-by-word recognition performed by MagiCoder, with the uncontrolled increase of the processed tokens (original words plus synonyms plus possible combinations), could induce a serious worsening of computational performances.

Thus, we moved from integrating word synonyms to add new locutions (i.e., phrases possibly composed by more words) as synonyms of an already existing MedDRA term. We recall that, unfortunately, the UMLS lexicon cannot help us in the generation of synonyms since it is not available in Italian. Therefore, we are exploring other solutions.

Up to now, we added to the official MedDRA terminology a set of about 1300 locutions. We automatically generated such a lexicon by considering three nouns that frequently occur in MedDRA, "aumento", "diminuzione" and "riduzione" (in English "increase", "decrease", and "reduction", respectively) and their adjectival form. For each LLT containing one of these nouns (resp., adjectives) we generated an equivalent term (also called pseudo-LLT) taking into account the corresponding adjective (resp., noun). A pseudo-LLT is regularly voted and sorted by MagiCoder but, if selected, the software returns the official (semantically equivalent) MedDRA term. Notice that, conversely to the single word synonyms solution, each pseudo-LLT is related to one and only one official term. This clearly limits the performance and computation time deterioration.

We repeated the first experiment on MagiCoder equipped with this small set of synonyms. The addition of synonyms induced a global improvement of MagiCoder performances on classes 4 and 5. For class 4, both precision and recall increased of 1%. For class 5, we observed a more significant increment as precision moved from 45% to 49%, while recall moved from 46% to 55%.

Class 5, which enjoys a particular advantage from the introduction of the pseudo-LLTs, represents a small part of reports. Notwithstanding, these cases are very arduous to address, and we have, at least, a good evidence of the effectiveness of our approach.

A different solution, working side-by-side with the pharmacovigilance experts, could consist in enlarging the MedDRA official terminology and generating a new ADR lexicon. This could be done on the basis of frequently retrieved locutions which are semantically equivalent to LLTs. "Candidate" pseudo-LLTs

can be retrieved adapting some statistical methods for linguistics [42], and then validated and linked to the official the terminology by experts. Some first experiments about the automatic generation of new MedDRA terminology can be found in [43].

5.3. Connectives in the narrative descriptions

As previously said, in MagiCoder we do not take into account the *structure* of sentences. From this point of view, our procedure is different from those based on the so called part-of-speech (PoS) [15]. PoS includes powerful methodologies able to perform the morpho-syntactical analysis of texts, labeling each lexical item with its grammatical properties. PoS-based text analyzers are also able to detect and deal with logical connectives such as conjunctions, disjunctions, and negations. Even if connectives generally play a central role in the logical foundation of natural languages, they have a minor relevance in the problem we are addressing. Indeed, ADR reports are, on average, badly/hurriedly written, or they do not have a complex structure (we empirically noted this also for long descriptions). Notwithstanding, negation deserves a distinct consideration, since the presence of a negation can drastically change the meaning of a phrase. We evaluated the frequency of negation connectives in ADR reports. We considered the same sample exploited in Section 4.1, and counted the occurrences of words "non" (Italian for "not") and "senza" (Italian for "without")[14]. We detected potential negations in 162 reports (i.e., only in the 3.6% of the total number, 4445). Even though negative sentences seem to be quite uncommon in ADR descriptions, the detection of negative forms is a short-term issue we plan to address. We are considering to adapt simple and efficient algorithm such as **NegEx** [44] (used, for example, in [33]) for identifying negations in textual medical records. **NegEx** has been developed for English language, it exploits a knowledge base (a "negation phrase list") and is able to distinguish different kinds of negation and to catch double negations. Some multilingual (Swedish, French and German) extensions of **NegEx** can be found in [45].

6. Conclusions and future work

In this paper we proposed MagiCoder, a simple and efficient NLP software, able to provide a concrete support to the pharmacovigilance task, in the revision of ADR spontaneous reports. MagiCoder takes in input a narrative description of a suspected ADR and produces as outcome a list of MedDRA terms that "cover" the medical meaning of the free-text description. Differently from other NLP software proposed in literature for the medical domain, we developed an original text processing procedure. Results of tests proposed in Section 4 about MagiCoder efficiency are encouraging.

[14] The word "senza" does not necessarily imply a negation, thus we are clearly overestimating the presence of negations.

As for ongoing and future work, we are addressing the inclusion of ad-hoc knowledge bases, as the MedDRA-thesaurus described in Section 5.2, and the study of suitable new heuristics and refinements to improve MagiCoder performances. We plan also to address the management of orthographic errors possibly contained in narrative ADR descriptions. A solution could be to include a medical term-oriented spell checker in VigiFarmaco or MagiCoder. In the first case, we would be able to prevent mistakes by pointing out to the user that she is doing some spelling error. This should drastically reduce users' orthographical errors without any side effect in MagiCoder development and performances. Note that a spell checker does not limit the user's expressiveness, but it just avoid trivial and common typos and it can be ignored when the user uses a word unknown to the spell checker. In the second case, MagiCoder would be able to recognize, and correct, spelling mistakes. Moreover, as already said, another short-term goal is to include in MagiCoder a subroutine for the detection and the management of negation connectives.

We are also providing evidence that MagiCoder is robust with respect to language and dictionary changes. The preliminary results in Section 4.3 suggest that MagiCoder can be a suitable tool also for narrative descriptions written in English. We plan to develop significative tests for MagiCoder on the English version of MedDRA and, moreover, we aim to test our procedure on different dictionaries (e.g., ICD[15], WHO-ART, SNOMED CT[16]).

Finally, we aim to apply MagiCoder to different sources of ADRs, such as drug information leaflets, clinical records, and social media [19, 46].

References

[1] N. Arthur, A. Bentsi-Enchill, R. Couper, et al., The Importance of Pharmacovigilance - Safety Monitoring of Medicinal Products, World Health Organization, 2002.

[2] European Medicines Agency, Guideline on good pharmacovigilance practices (GVP) - Module VI, EMA/873138/2011 (2014).

[3] International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use (ICH), MedDRA data retrieval and presentation: points to consider (2016).

[4] D. Jurafsky, J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 1st Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[15] http://www.who.int/classifications/icd/en/
[16] https://www.snomed.org/snomed-ct

[5] G. Attardi, V. Cozza, D. Sartiano, Annotation and extraction of relations from Italian medical records, in: Proceedings of the 6th Italian Information Retrieval Workshop, Cagliari, Italy, 2015.

[6] M. Zorzi, C. Combi, R. Lora, M. Pagliarini, U. Moretti, Automagically encoding adverse drug reactions in MedDRA, in: P. Balakrishnan, J. Srivatsava, W. Fu, S. M. Harabagiu, F. Wang (Eds.), 2015 IEEE International Conference on Healthcare Informatics, ICHI 2015, Dallas, TX, USA, October 21-23, 2015, IEEE Computer Society, 2015, pp. 90–99. doi:10.1109/ICHI.2015.18.

[7] J. Borg, G. Aislaitner, M. Pirozynski, S. Mifsud, Strengthening and rationalizing pharmacovigilance in the EU: where is Europe heading to? A review of the new EU legislation on pharmacovigilance, Data Safety 34 (3) (2011) 187–197.

[8] L. Aagaard, J. Strandell, L. Melskens, P. Petersen, E. Holme Hansen, Global patterns of adverse drug reactions over a decade: analyses of spontaneous reports to VigiBase, Drug Safety 35 (2012) 1171–1182.

[9] P. Radhakrishna, Upversioning MedDRA dictionary - insights from a seasoned coder, Data Basics 20 (3) (2014) 1171–1182.

[10] A. Bate, S. Evans, Quantitative signal detection using spontaneous ADR reporting, Pharmacoepidemiology and Drug Safety 18 (6) (2009) 427–436.

[11] X. Wang, G. Hripcsak, M. Markatou, C.Friedman, Active computerized pharmacovigilance using natural language processing, statistics, and electronic health records: A feasibility study, JAMIA 16 (3) (2009) 328–337.

[12] C. Friedman, Discovering novel adverse drug events using natural language processing and mining of the electronic health record, in: Artificial Intelligence in Medicine, Vol. 5651 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 1–5.

[13] E. Aramaki, Y. Miura, M. Tonoike, T. Ohkuma, H. Masuichi, K. Waki, Extraction of adverse drug effects from clinical records, Stud Health Technol Inform 160 (Pt1) (2010) 739–743.

[14] M. G. R. Reichley, P. Kilbridge, L. Noirot, R. N. R, W. Dunagan, T. Bailey, Natural language processing to identify adverse drug events, in: AMIA Annu Symp Proc., 2008.

[15] F. Dell'Orletta, Ensemble system for part-of-speech tagging, in: Proceedings of EVALITA - Evaluation of NLP and Speech Tools for Italian, Reggio Emilia, Italy, 2009, pp. 1–6.

[16] M. Cristani, A. Bertolaso, S. Scannapieco, C. Tomazzoli, Future paradigms of automated processing of business documents, International Journal of Information Management 40 (2018) 67–75.

[17] H. Gurulingappa, A. Mateen-Rajput, L. Toldo, Extraction of potential adverse drug events from medical case reports, Journal of Biomedical Semantics 3 (15) (2012) 1–10.

[18] A. Sarker, G. Gonzalez, Portable automatic text classification for adverse drug reaction detection via multi-corpus training, Journal of Biomedical Informatics 53 (2015) 196–207.

[19] C. C. Yang, H. Yang, L. Jiang, M. Zhang, Social media mining for drug safety signal detection, in: Proc. of the 2012 Int. Workshop on Smart Health and Wellbeing, SHB 2012, 2012, pp. 33–40.

[20] G. Dalhberg, Implementation and evaluation of a text extraction tool for adverse drug reaction information, Master's thesis, Uppsala University School of Engineering (2010).

[21] M. F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.

[22] M. Collins, Tutorial: Machine learning methods in natural language processing, in: 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Vol. 2777 of Lecture Notes in Computer Science, Springer, Washington, DC, USA, 2003, p. 655.

[23] A. Brayfield, Martindale: the complete drug reference, PhP, Pharmaceutical Press, London, UK, 2014.

[24] A. Coffman, N. Wharton, Clinical natural language processing: autoassigning ICD-9 codes, in: Overview of the Computational Medicine Center's 2007 Medical Natural Language Processing Challenge. Available online at http://courses.ischool.berkeley.edu/i256/f09/FinalProjects

[25] B. Ribeiro-Neto, A. H. Laender, L. R. de Lima, An experimental study in automatically categorizing medical documents, Journal of the American Society for Information Science and Technology 52 (5) (2001) 391–401.

[26] J. van der Zwaan, E. Tjong Kim Sang, M. de Rijke, An experiment in automatic classification of pathological reports, in: R. Bellazzi, A. Abu-Hanna, J. Hunter (Eds.), Proceedings of the 11th Conference on Artificial Intelligence in Medicine, AIME 2007, Springer Berlin Heidelberg, Amsterdam, The Netherlands, 2007, pp. 207–216.

[27] A. Prez, K. Gojenola, A. Casillas, M. Oronoz, A. D. de Ilarraza, Computer aided classification of diagnostic terms in Spanish, Expert Systems with Applications 42 (6) (2015) 2949 – 2958.

[28] U. Hahn, M. Romacker, S. Schulz, medSynDiKATe - a natural language system for the extraction of medical information from findings reports, International Journal of Medical Informatics 67 (13) (2002) 63 – 74.

[29] D. Demner-Fushman, W. W. Chapman, C. J. McDonald, What can natural language processing do for clinical decision support?, Journal of Biomedical Informatics.

[30] J. Banda, L. Evans, R. Vanguri, N. Tatonetti, P. Ryan, N. Shah, An algorithm for suffix stripping, A curated and standardized adverse drug event resource to accelerate drug safety research. 3:160026.

[31] R. Harpaz, H. S. Chase, C. Friedman, Mining multi-item drug adverse effect associations in spontaneous reporting systems, BMC Bioinformatics 11 (S-9) (2010) S7.

[32] N. Nissim, M. Boland, R. Moskovitch, N. Tatonetti, Y. Elovici, Y. Shahar, G. Hripcsak, An active learning framework for efficient condition severity classification, in: Artificial Intelligence in Medicine (AIME'15), Vol. 9105 of Lecture Notes in Computer Science, Springer, 2015, pp. 13–24.

[33] S. Meystre, P. J. Haug, Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation, J Biomed Inform 39 (6) (2006) 589 – 599.

[34] M. Cristani, F. Olivieri, C. Tomazzoli, M. Zorzi, Towards a logical framework for diagnostic reasoning, in: Proceedings of 12th KES Conference on Agent and Multi-Agent Systems: Technologies and Applications, KES-AMSTA-18, Smart Innovation, Systems and Technologies, Volume 96, 2018, Springer, 2018, pp. 144–155. doi:https://doi.org/10.1007/978-3-319-92031-3_14.

[35] L. Bauer, Introducing linguistic morphology, Edinburgh University Press, Edinburgh, 2003.

[36] K. Kishida, Technical issues of cross-language information retrieval: A review, Inf. Process. Manage. 41 (3) (2005) 433–455.

[37] A. Clark, C. Fox, S. Lappin (Eds.), The Handbook of Computational Linguistics and Natural Language Processing, Blackwell Handbooks in Linguistics, John Wiley & Sons, 2010.

[38] J. Piskorski, M. M. Sydow, String distance metrics for reference matching and search query correction, in: W. Abramowicz (Ed.), Business Information Systems, Vol. 4439 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 353–365.

[39] A. Sabaini, Temporal data analysis and mining: A multidimensional approach and its application in a medical domain, Ph.D. thesis, Department of Computer Science, University of Verona - Italy (2015).

[40] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.

[41] J. Savoy, Report on CLEF-2001 experiments, Tech. rep., Université de Neuchâtel, Switzerland (2001).

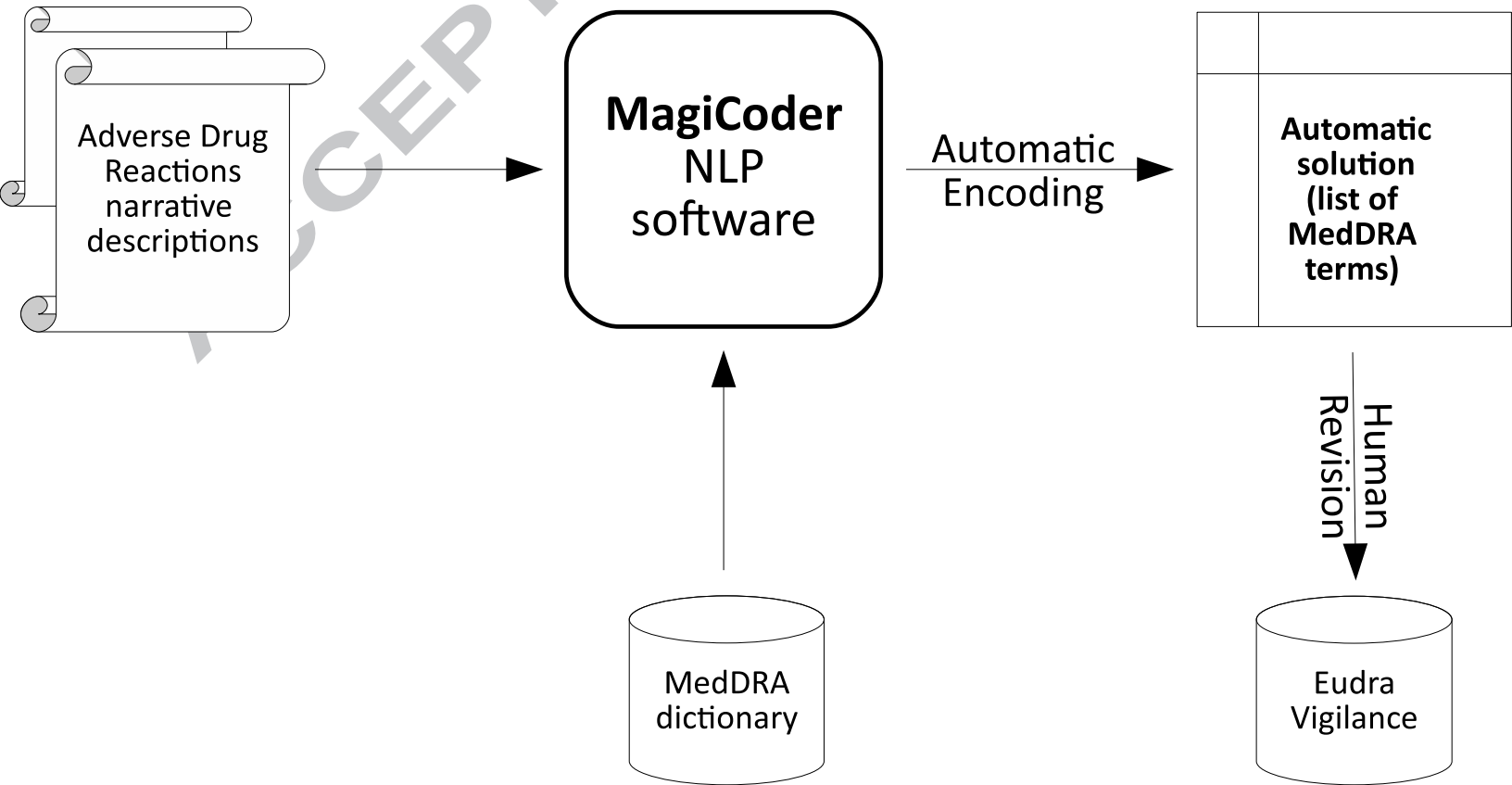
[42] M. Baroni, S. Bisi, Using cooccurrence statistics and the web to discover synonyms in a technical language, in: Proc. of LREC, 2004.

[43] M. Zorzi, C. Combi, G. Pozzani, E. Arzenton, U. Moretti, A co-occurrence based MedDRA terminology generation: Some preliminary results, in: Proceedings of the 16th Conference on Artificial Intelligence in Medicine, AIME 2017, Vol. 10259 of Lecture Notes in Computer Science, Springer, Vienna, Austria, 2017, pp. 215–220.

[44] W. W. Chapman, W. Bridewell, P. Hanbury, G. F. Cooper, B. G. Buchanan, A simple algorithm for identifying negated findings and diseases in discharge summaries, Journal of Biomedical Informatics 34 (5) (2001) 301 – 310.

[45] W. W. Chapman, D. Hillert, S. Velupillai, M. Kvist, M. Skeppstedt, B. E. Chapman, M. Conway, M. Tharp, D. L. Mowery, L. Deléger, Extending the negex lexicon for multiple languages, in: MEDINFO 2013 - Proceedings of the 14th World Congress on Medical and Health Informatics, Copenhagen, Denmark, 2013, pp. 677–681.

[46] A. Sarker, R. Ginn, A. Nikfarjam, K. O'Connor, K. Smith, S. Jayaraman, T. Upadhaya, G. Gonzalez, Utilizing social media data for pharmacovigilance: A review, Journal of Biomedical Informatics 54 (2015) 202 – 212.



1) The NLP software MagiCoder, that automatically maps spontaneous reports into MedDRA terminology is introduced.

2) We tested MagiCoder against a gold standard of about 1800 manually revised reports.
We measured an average recall and precision of 86, 9% and 91,8%, respectively.

3) We also performs some initial, encouraging experiments on English texts.

4) From a practical point of view, MagiCoder reduces the time required for encoding ADR reports. This improvement in the efficiency of pharmacologists' work has a relevant impact also on the quality of the subsequent data analysis.