From narrative descriptions to MedDRA: automagically encoding adverse drug reactions

Abstract

The collection of narrative spontaneous reports is an irreplaceable source for the prompt detection of suspected adverse drug reactions (ADRs): 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. MagiCoder, a Natural Language Processing algorithm, is proposed for the automatic encoding of free-text descriptions into MedDRA terms. MagiCoder procedure is efficient in terms of computational complexity (in particular, it is linear in the size of the narrative input and the terminology). We tested it on a large dataset of about 4500 manually revised reports, by performing an automated comparison between human and MagiCoder revisions. For the current base version of MagiCoder, we measured: on short descriptions, an average recall of 86% and an average precision of 88%; on medium-long descriptions (up to 255 characters), an average recall of 64% and an average precision of 63%. From a practical point of view, MagiCoder reduces the time required for encoding ADR reports. Pharmacologists have simply to review and validate the MagiCoder terms proposed by the application, instead of choosing the right terms among the 70K low level terms of MedDRA. Such improvement in the 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.

Introduction

Pharmacovigilance includes all activities aimed to systematically study risks and benefits related to the

correct use of marketed drugs. The development of a new drug, which begins with the production and ends with the commercialization of a pharmaceutical product, considers both pre-clinical studies (usually tests on animals) and clinical studies (tests on patients). After these phases, a pharmaceutical company can require the authorization for the commercialization of the new drug. Notwithstanding, whereas at this stage drug benefits are well-know, results about drug safety are not conclusive BIBREF0 . The pre-marketing tests cited above have some limitations: they involve a small number of patients; they exclude relevant subgroups of population such as children and elders; the experimentation period is relatively short, less than two years; the experimentation does not deal with possibly concomitant pathologies, or with the concurrent use of other drugs. For all these reasons, non-common 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 commercialization. It may happen that drugs are withdrawn from the market after the detection of unexpected collateral 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 BIBREF0 .

Spontaneous reporting is the main method pharmacovigilance adopts in order to identify adverse drug reactions. Through spontaneous reporting, health care professionals, patients, and pharmaceutical companies can voluntarily send information about suspected ADRs to the national regulatory authority. The spontaneous reporting is an important activity. It provides pharmacologists and regulatory authorities with early alerts, by considering every drug on the market and every patient category.

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 National Network of Pharmacovigilance (RNF, in Italian) within seven days since they have been received. Once reports have been notified and sent to RNF they are analysed by both local pharmacovigilance centres and by the Drug Italian Agency (AIFA).

Subsequently, they are sent to Eudravigilance BIBREF1 and to VigiBase BIBREF2 (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 citizens. In last years, the number of ADR reports in Italy has grown rapidly, going from approximately ten thousand in 2006 to around sixty thousand in 2014 BIBREF3 , as shown in Figure FIGREF3 .

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 the reports became difficult and often unbearable both by people responsible for pharmacovigilance and by regional centres. Indeed, each report must be checked, in order to control its quality; it is consequently encoded and transferred to RNF via “copy by hand” (actually, a printed copy).

Recently, to increase the efficiency in collecting and managing ADR reports, a web application, called VigiFarmaco, has been designed and implemented for the Italian pharmacovigilance. Through VigiFarmaco, a spontaneous report can be filled out online by both healthcare professionals and citizens (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 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 files BIBREF4) to RNF by simply clicking a button, after reviewing it.

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. Notwithstanding, there is an operation which still requires the manual intervention of responsibles for pharmacovigilance also for online report revisions: the encoding in MedDRA terminology of the free text, through which the reporter describes one or more adverse drug reactions. MedDRA (Medical Dictionary for Regulatory Activities) is a medical terminology introduced with the purpose to standardize and facilitate the sharing of information about medicinal products in particular with respect to regulatory activities BIBREF5 . 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 citizens, 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.

According to the described scenario, in this paper we propose INLINEFORM0 , an original Natural Language Processing (NLP) BIBREF6 algorithm and related software tool, which automatically assigns one or more terms from a dictionary to a narrative text. A preliminary version of INLINEFORM1 has been proposed in BIBREF7 . MagiCoder has been first developed for supporting pharmacovigilance supervisors in using VigiFarmaco, providing them with an initial automatic MedDRA encoding of the ADR descriptions in the online reports collected by VigiFarmaco, that the supervisors check and may correct or accept as it 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 free narrative text. With respect to other solutions already available in literature and market, MagiCoder has been designed to be efficient and less computationally expensive, unsupervised, and with no need of training. MagiCoder uses stemming to be independent from 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 first version BIBREF7 , we extended our proposal following several directions. First of all, we refined the procedure: MagiCoder has been equipped with some heuristic criteria and we started to address the problem of including auxiliary dictionaries (e.g., in order to deal with synonyms). MagiCoder computational complexity has been carefully studied and we will show that it is linear in the size of the dictionary (in this case, the number of LLTs in MedDRA) and the text description. We performed an accurate test of MagiCoder performances: by means of well-known statistical measures, we collected a significant set of quantitative information about the effective behavior of the procedure. We

largely discuss some crucial key-points we met in the development of this version of MagiCoder, proposing short-time solutions we are addressing as work in progress, such as changes in stemming algorithm, considering synonyms, term filtering heuristics.

The paper is organized as follows. In Section SECREF2 we provide some background notions and we discuss related work. In Section SECREF3 we present the algorithm MagiCoder, by providing both a qualitative description and the pseudocode. In Section SECREF4 we spend some words about the user interface of the related software tool. In Section SECREF5 we explain the benchmark we developed to test INLINEFORM0 performances and its results. Section SECREF6 is devoted to some discussions. Finally, in Section SECREF7 we summarize the main features of our work and sketch some future research lines.

Natural language processing and text mining in medicine

Automatic detection of adverse drug reactions from text has recently received an increasing interest in pharmacovigilance research. Narrative descriptions of ADRs come from heterogeneous sources: spontaneous reporting, Electronic Health Records, Clinical Reports, and social media. In BIBREF8 , BIBREF9 , BIBREF10 , BIBREF11 , BIBREF12 some NLP approaches have been proposed for the extraction of ADRs from text. In BIBREF13 , the authors collect narrative discharge summaries from the Clinical Information System at New York Presbyterian Hospital. MedLEE, an NLP 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 BIBREF14 , the authors 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 BIBREF15 , where the authors 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 DailyStrenght, 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 ADR description task. In BIBREF16 the authors propose to use association mining and Proportional Reporting Ratio (PRR, a well-know pharmacovigilance statistical index) to mine the 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 health care communities), the authors apply the Consumer Health Vocabulary to generate ADR lexicon. ADR lexicon is a computerized collection 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 SECREF12). 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.

Other related papers about pharmacovigilance and machine learning or data mining are BIBREF17 , BIBREF18 . In BIBREF19 , a text extraction tool is implemented on the .NET platform for preprocessing text (removal of stop words, Porter stemming BIBREF20 and use of synonyms) and matching medical terms using permutations of words and spelling variations (Soundex, Levenshtein distance and Longest common subsequence distance BIBREF21). Its performance has been evaluated on both manually extracted medical terms from summaries of product characteristics and unstructured adverse effect texts from Martindale (a medical reference for information about drugs and medicines) using the WHO-ART and MedDRA medical terminologies. A lot of linguistic features have been considered and a careful analysis of performances has been provided. In BIBREF22 the authors develop an algorithm in order to

help coders in the subtle task of auto-assigning ICD-9 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 INLINEFORM0 .

MedDRA Dictionary

The Medical Dictionary for Regulatory Activities (MedDRA) BIBREF5 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 BIBREF23 . 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 Mainteinance And Service Organization (MSSO). 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 TABREF8 shows an example of the hierarchy: reaction Itch is described starting from Skin disorders (SOC), Epidermal conditions (HLGT), Dermatitis and Eczema (HLT), and Asteatotic Eczema (PT). Preferred Terms are Low Level Terms chosen to be representative of a group of terms. It should be stressed that the hierarchy is multiaxial: for example, a PT can be grouped into one or more HLT, but it belongs to only one primary SOC term.

MagiCoder: an NLP software for ADR automatic encoding

A natural language ADR description is a completely free text. The user has no limitations, she/he can potentially write everything: a number of online ADR descriptions actually contain 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 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 SECREF66) or the naive ad hoc management of particular cases, can limit the efficiency and the desired of the algorithm. For these reasons, we carefully designed INLINEFORM0 , 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 BIBREF24 , BIBREF6 , BIBREF25 . According to the described pharmacovigilance domain, we considered algorithms for the morpho-analysis and the part-of-speech (PoS) extraction techniques BIBREF24 , BIBREF6 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 LLTs. This task has to be done in a very 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:

In this paper we consider the Italian context of Pharmacovigilance and, as a consequence, we will consider and process by MagiCoder textual descriptions written in Italian language. We will discuss the potentiality of MagiCoder on other languages and some preliminary results in Section SECREF7 .

MagiCoder: overview

The main idea of INLINEFORM0 is that a single linear scan of the free-text is sufficient, in order to recognize INLINEFORM1 terms.

From an abstract point of view, we try to recognize, in the narrative description, single words belonging to LLTs, which 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 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 BIBREF19).

We can distinguish five phases in the procedure that will be discussed in detail in Sections UID18 , UID19 , UID20 , UID23 , UID28 , respectively.

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.

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.

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.

Weights calculation: recognized terms are weighted depending on information about syntactical matching.

Sorting of voted terms and winning terms release: the set of voted term is pruned, terms are sorted and finally a solution (a set of winning terms) is released.

The algorithm proceeds with a word-by-word comparison. We iterate on the preprocessed text and we

test if a single word INLINEFORM0 , 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. 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 SECREF48 , in INLINEFORM0 time units (where INLINEFORM1 and INLINEFORM2 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 INLINEFORM3 contains information about the set of LLTs containing INLINEFORM4 . This way, we can verify the presence in MedDRA of a word INLINEFORM5 encountered in the ADR description in constant time. We call this meta-dictionary INLINEFORM6 . We build a meta dictionary also from a stemmed version of MedDRA, to verify the presence of stemmed descriptions. We call it INLINEFORM7 . 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 SECREF7 .

Given a natural language ADR description, the text has to be preprocessed in order to perform an efficient computation. We adopt a well-know technique such as tokenization BIBREF26 : 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 SECREF6 .

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 BIBREF20 , BIBREF26 . 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 positives” 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 SECREF6 .

INLINEFORM0 scans the text word-by-word (remember that each word corresponds to a token) once and performs a “voting task”: at the INLINEFORM1 -th step, it marks (i.e., “votes”) with index INLINEFORM2 each LLT INLINEFORM3 containing the current (INLINEFORM4 -th) word of the ADR description. Moreover, it keeps track of the position where the INLINEFORM5 -th word occurs in INLINEFORM6 .

INLINEFORM0 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 INLINEFORM1 has been exactly recognized in a term INLINEFORM2 , the match between the stemmed versions of INLINEFORM3 and INLINEFORM4

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 INLINEFORM5 the sub-dictionary of voted terms.

Each voted term INLINEFORM0 is equipped with two auxiliary data structures, containing, respectively:

the positions of the voting words in the ADR description; we call INLINEFORM0 this sequence of indexes;

the positions of the voted words in the MedDRA term INLINEFORM0 ; we call INLINEFORM1 this sequence of indexes.

Moreover, we endow each voted term INLINEFORM0 with a third structure that will contain the sorting criteria we define below; we will call it INLINEFORM1 .

Let us now introduce some notations we will use in the following. We denote as INLINEFORM0 the function that, given an LLT INLINEFORM1 , returns the number of words contained in INLINEFORM2 (excluding the stop words). We denote as INLINEFORM3 (resp. INLINEFORM4) the function that returns the number of indexes belonging to INLINEFORM5 (resp. INLINEFORM6). We denote as INLINEFORM7 and INLINEFORM8 the functions that return the maximum and the minimum indexes in INLINEFORM9 , 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,

INLINEFORM0 will include, among others, “Anaphylactic shock” (10002199) and “Anaphylactic reaction to drug” (10054844). We will have that INLINEFORM1 (i.e., “anaphylactic” and “shock”) while INLINEFORM2 (i.e., “anaphylactic” and “drug”). On the other hand, INLINEFORM3 , revealing that both words in the term have been voted, while INLINEFORM4 , 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.

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 assigns 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 UID28).

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

In the following, INLINEFORM0 is a normalization factor (w.r.t. the length, in terms of words, of the LLT INLINEFORM1). 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 worst cases.

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

INLINEFORM0

In the example we introduced before, we have that INLINEFORM0 (i.e., all words of the terms have been recognized in the description) while INLINEFORM1 (i.e., one word out of three has not been recognized in the description).

The algorithm considers whether a perfect matching has been performed using or not stemmed words. INLINEFORM0 is simply a flag. INLINEFORM1 holds if stemming has been used at least once in the voting procedure of INLINEFORM2 , and it is valued 1, otherwise it is valued 0.

For example, INLINEFORM0 and INLINEFORM1 .

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 BIBREF27 , which is robust with respect to word permutation. Thus, INLINEFORM0

where INLINEFORM0 is the pair distance function (between strings INLINEFORM1 and INLINEFORM2) and INLINEFORM3 is the term “rebuilt” from the words in ADR description corresponding to indexes in INLINEFORM4 .

For example, INLINEFORM0 (i.e., the concatenation of the voters and the term are equal) and INLINEFORM1 .

We want to estimate how an LLT has been covered. INLINEFORM0

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 INLINEFORM0 we have the data structure INLINEFORM1 , 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 the example introduced before, we have that INLINEFORM0 while INLINEFORM1 . Thus, concluding, we obtain that INLINEFORM2 while INLINEFORM3 .

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 onlt best suitable solutions. We will provide an 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 criterium.

As a first step, we provide an initial pruning of the syntactically retrieved terms guided by the ordered-phrases heuristic criterium. In the ordered-phrases criterium we reintroduce the order of words in the narrative description as a selection discriminating factor. From the set of selected LLTs, 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 corresponing 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 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 criterium 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 INLINEFORM0 the set of voted terms after the selection by the ordered-phrases criterium. We proceed then by ordering INLINEFORM1 : we use a multiple-value sorting on elements in INLINEFORM2 , for each INLINEFORM3 . The obtained subdictionary is dubbed as INLINEFORM4 and it has possibly 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 INLINEFORM0 as follows. We iterate on the ordered dictionary and for each INLINEFORM1 we select INLINEFORM2 if all the following conditions hold:

INLINEFORM0 is completely covered, i.e., INLINEFORM1 ;

INLINEFORM0 does not already belong to INLINEFORM1 ;

INLINEFORM0 is not a prefix of another selected term INLINEFORM1 ;

INLINEFORM0 has been voted without stemming (i.e., INLINEFORM1) or, for any INLINEFORM2 , INLINEFORM3 has not been covered (i.e., none term voted by INLINEFORM4 has been already selected) or INLINEFORM5 has not been exactly covered (i.e., only its stem has been recognized in some term INLINEFORM6).

At this stage, we have a set of MedDRA terms which “covers” the narrative description. We further select a subset INLINEFORM0 of INLINEFORM1 with a second heuristic, the maximal-set-of-voters criterium.

The maximal-set-of-voters criterium deletes from the solution those terms which can be considered “extensions” of other ones. For each pair of terms INLINEFORM0 and INLINEFORM1 , it checks if INLINEFORM2 is a subset of INLINEFORM3 (considered as sets of indexes). If it is the case, INLINEFORM4 is removed from INLINEFORM5 .

In INLINEFORM0 we do not need to consider ad hoc subroutines to address permutations and combinations of words (as it is done, for example, in BIBREF19). 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 the 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 INLINEFORM1 , but the order between words does not necessarily matter.

MagiCoder: structure of the algorithm

Figure SECREF34 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 INLINEFORM0 takes the narrative description, performs tokenization and stop-word removal and puts it into an array of words.

Procedures INLINEFORM0 and INLINEFORM1 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 INLINEFORM0 (resp., INLINEFORM1), we refer to the set of LLTs containing the word INLINEFORM2 (resp., the stem of INLINEFORM3).

Function INLINEFORM0 returns the stemmed version of word INLINEFORM1 .

Function INLINEFORM0 returns the position of word INLINEFORM1 in term INLINEFORM2 .

INLINEFORM0 is a flag, initially set to 0, which holds 1 if at least a stemmed matching with the MedDRA term INLINEFORM1 is found.

INLINEFORM0 , INLINEFORM1 , INLINEFORM2 are arrays and INLINEFORM3 appends INLINEFORM4 to array INLINEFORM5 , where INLINEFORM6 may be an element or a sequence of elements.

INLINEFORM0 (INLINEFORM1) are the weights related to the criteria defined in Section UID23 .

Procedure INLINEFORM0 performs the multi-value sorting of the array INLINEFORM1 based on the

values of the properties INLINEFORM2 of its elements.

Procedure INLINEFORM0 , where INLINEFORM1 is a set of terms and INLINEFORM2 is a term, tests whether INLINEFORM3 (considered as a string) is prefix of a term in INLINEFORM4 . Dually, procedure INLINEFORM5 tests if in INLINEFORM6 there are one or more prefixes of INLINEFORM7 , and eventually remove them from INLINEFORM8 .

Function INLINEFORM0 specifies whether a word INLINEFORM1 has been already covered (i.e., a term voted by INLINEFORM2 has been selected) in the (partial) solution during the term release: INLINEFORM3 holds 1 if INLINEFORM4 has been covered (with or without stemming) and it holds 0 otherwise. We assume that before starting the final phase of building the solution (i.e., the returned set of LLTs), INLINEFORM5 for any word INLINEFORM6 belonging to the description.

Procedures INLINEFORM0 and INLINEFORM1 , where INLINEFORM2 is a set of terms, implement ordered-phrases and maximal-set-of-voters criteria (defined in Section UID28), respectively.

Function INLINEFORM0 , returns the first INLINEFORM1 elements of an ordered set INLINEFORM2 . If INLINEFORM3 , the function returns the complete list of ordered terms and INLINEFORM4 nil values.

[!t] MagiCoder(INLINEFORM0 text, INLINEFORM1 dictionary, INLINEFORM2 integer)

INLINEFORM0 : the narrative description;

INLINEFORM0 : a data structure containing the MedDRA INLINEFORM1 s;

INLINEFORM0 : the maximum number of winning terms that have to be released by the procedure an

ordered set of LLTs INLINEFORM1 = CreateMetaDict(INLINEFORM2) INLINEFORM3 = CreateStemMetaDict(INLINEFORM4) adr_clear = Preprocessing(INLINEFORM5) adr_length = adr_clear.length INLINEFORM6 = INLINEFORM7 for each non-stop-word in the description (i INLINEFORM8 test whether the current word belongs to MedDRA adr_clear[i] INLINEFORM9 for each term containing the word t INLINEFORM10 (adr_clear[i]) keep track of the index of the voting word INLINEFORM11 [INLINEFORM12 ,i] keep track of the index of the recognized word in INLINEFORM13 INLINEFORM14 [INLINEFORM15 , INLINEFORM16 (adr_clear[i])]

INLINEFORM0 = INLINEFORM1 test if the current (stemmed) word belongs the stemmed MedDRA stem(adr_clear[i]) INLINEFORM2 t INLINEFORM3 (stem(adr_clear[i])) test if the current term has not been exactly voted by the same word i INLINEFORM4 INLINEFORM5 [INLINEFORM6 , i] INLINEFORM7 [INLINEFORM8 , INLINEFORM9 (adr_clear[i])] keep track that INLINEFORM10 has been covered by a stemmed word INLINEFORM11 = true INLINEFORM12 = INLINEFORM13 for each voted term, calculate the four weights of the corresponding criteria t INLINEFORM14 INLINEFORM15 [INLINEFORM16] filtering of the voted terms by the first heuristic criterium INLINEFORM17 multiple value sorting of the voted terms INLINEFORM18 = sortby(INLINEFORM19) t INLINEFORM20 index INLINEFORM21 select a term INLINEFORM22 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 INLINEFORM23 and if INLINEFORM24 is not prefix of another already selected terms INLINEFORM25 AND ((INLINEFORM26 = false OR (mark(adr_clear(index))=0)) AND t INLINEFORM27 AND prefix(INLINEFORM28 ,t)=false) mark(adr_clear(index))=1 remove from the selected term set all terms which are prefix of INLINEFORM29 INLINEFORM30 = remove_prefix(INLINEFORM31 ,t) INLINEFORM32 = INLINEFORM33 filtering of the finally selected terms by the second heuristic criterium INLINEFORM34 INLINEFORM35 INLINEFORM36

Pseudocode of MagiCoder

MagiCoder complexity analysis

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

Let INLINEFORM0 be the input size (the length, in terms of words, of the narrative description). Let INLINEFORM1 be the cardinality of the dictionary (i.e., the number of terms). Moreover, let INLINEFORM2 be the number of distinct words occurring in the dictionary and let INLINEFORM3 be the length of the longest term in the dictionary. For MedDRA, we have about 75K terms (INLINEFORM4) and 17K unique words (INLINEFORM5). Notice that, reasonably, INLINEFORM6 is a small constant for any dictionary; in particular, for MedDRA we have INLINEFORM7 . We assume that all update operations on auxiliary data structures require constant time INLINEFORM8 .

Building meta-dictionaries INLINEFORM0 and INLINEFORM1 requires INLINEFORM2 time units. In fact, the simplest procedure to build these hash tables is to scan the LLT dictionary and, for each term INLINEFORM3 , to verify for each word INLINEFORM4 belonging to INLINEFORM5 whether INLINEFORM6 is a key in the hash table (this can be done in constant time). If INLINEFORM7 is a key, then we have to update the values associated to INLINEFORM8 , i.e., we add INLINEFORM9 to the set of terms containing INLINEFORM10 . Otherwise, we add the new key INLINEFORM11 and the associated term INLINEFORM12 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 them.

It can be easily verified that the voting procedure requires in the worst case INLINEFORM0 steps: 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 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. Thus, usually, real computational complexity is much less of this worst case.

The computation of criteria-related weights requires INLINEFORM0 time units. In particular: both criterion one and criterion two require INLINEFORM1 time steps; criterion three require INLINEFORM2 (we assume to absorb the complexity of the pair distance function); criterion four requires INLINEFORM3 time units.

The subsequent multi-value sorting based on computed weights is a sorting algorithm which complexity can be approximated to INLINEFORM0 , 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 multi-sorting is constant, it can be neglected. Thus, the complexity of multi-value sorting can be considered to be INLINEFORM1 .

Finally, to derive the best solutions actually requires INLINEFORM0 steps. The ordered-phrases criterium requires INLINEFORM1 ; the maximal set of voters criterium takes INLINEFORM2 time units.

Thus, we conclude that MagiCoder requires in the worst case INLINEFORM0 computational steps. We again highlight that this is a (very) worst case scenario, while in average it performs quite better. Moreover, we did not take into account that each phase works on a subset of terms of the previous phase, and the size of these subset rapidly decreases in common application.

the selection phase works only on voted terms, thus, in common applications, on a subset of the original dictionary.

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 FIGREF50 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 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: the responsible 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 results 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.

Testing MagiCoder performances

In this section we describe the experiments we performed to evaluate MagiCoder performances. The test exploits a large amount of manually revised reports we obtained from VigiSegn BIBREF3 .

We briefly recall two metrics we used to evaluate MagiCoder: precision and recall.

In statistical hypothesis and in particular in binary classification BIBREF28 , two main kinds of errors are pointed out: false positive errors (FP) and false negative errors (FN). In our setting, these errors can be viewed as follows: a false positive error is the inopportune retrieval of a “wrong” LLT, i.e., a term which

does not correctly encode the textual description; a false negative error is the failure in the recognition of a “good” LLT, i.e., a term which effectively encode (a part of) the narrative description and that would have been selected by a human expert. As dual notions of false positive and false negative, one can define correct results, i.e., true positive (TP) and true negative (TN): in our case, a true positive is a correctly returned LLT, and a true negative is an LLT which, correctly, has not been recognized as a solution.

Following the information retrieval tradition, the standard approach to system evaluation revolves around the notion of relevant and non-relevant solution (in information retrieval, a solution is represented by a document BIBREF28). We provide here a straightforward definition of relevant solution. A relevant solution is a MedDRA term which correctly encode the narrative description provided to MagiCoder. A retrieved solution is trivially defined as an output term, independently from its relevance. We dub the sets of relevant solutions and retrieved solutions as INLINEFORM0 and INLINEFORM1 , respectively.

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.

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 returned by the system.

Table TABREF51 summarizes formulas for precision and recall. We provide formulas both in terms of relevant/retrieved solutions and false positives, true positives and false negatives.

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 has to be represented by a human encoding of a narrative description, i.e., a set of MedDRA terms choosen by a pharmacovigilance expert. Such a set is assumed to be definitively correct (only correct solutions are returned) and complete (all correct solutions have been returned).

Experiment about MagiCoder performances

To evaluate MagiCoder performances, we 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 BIBREF3 . This system is based on the open source business intelligence suite Pentaho. VigiSegn offers a large number of encoded ADRs. The encoding has been manually performed and validated by experts working at pharmacovigilance centres. Encoding results have then been sent to the national regulatory authority, AIFA.

We performed a test composed by the following steps.

We launch an ETL procedure through Pentaho Data Integration. Reports are transferred from VigiSegn to an ad hoc database TestDB. The dataset covers all the 4445 reports received, revised and validated during the year 2014 for the Italian region Veneto.

The ETL procedure extracts the narrative descriptions from reports stored in TestDB. For each description, the procedure calls MagiCoder from

VigiFarmaco; the output, i.e., a list of MedDRA terms, is stored in a table of TestDB.

Manual and automatic encodings of each report are finally compared through an SQL query. 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. Moreover, we map each LLT term recognized by both the human experts and MagiCoder to its corresponding preferred term. Results are discussed below in Section UID57 .

Table TABREF58 shows the results of this first performance test. We group narrative descriptions by increasing length (in terms of characters). We note that reported results are computed considering terms at PT level. 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 LLTs 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 BIBREF5 . With common PT we mean the percentage of preferred terms retrieved by human reviewers that have been recognized also by MagiCoder. Reported performances are summarized also in FIGREF59 . Note that, false positive and false negative errors are required to be as small as possible, while common PT, recall, and precision have to be as large as possible.

MagiCoder behaves very well on very short descriptions (class 1) and on short ones (class 2). Recall and precision remain greater than 50% up to class 4. Notice that very long descriptions (class 5), on which performances drastically decrease, represent a negligible percentage of the whole set (less than 0.3%). Some remarks are mandatory. It is worth noting that this test simply estimates how much, for each report, the MagiCoder behavior is similar to the manual work, without considering the effective quality of the manual encoding. Clearly, as a set of official reports, revised and sent to RNF, we assume to deal with an high-quality encoding: notwithstanding, some errors in the human encoding possibly occur. Moreover, 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 make some encoding choices in order to avoid repetitions. Thus, an LLT INLINEFORM0 returned by MagiCoder that has not been selected by the expert because redundant is not truly a false positive. As a significative counterpart, as previously said, we notice that some reports contain slightly human omissions/errors. This suggest the evidence that we are underestimating MagiCoder performances. See the next section for some simple but significative examples.

Examples

Table TABREF61 provides some examples of the behavior of MagiCoder. We propose some free-text ADR descriptions from TestDB and we provide both the manual and the automatic encodings into LLT terms. We also provide the English translation of the natural language texts (we actually provide a quite straightforward literal translation).

In Table TABREF61 we use the following notations: INLINEFORM0 and INLINEFORM1 are two identical LLTs retrieved both by the human and the automatic encoding; INLINEFORM2 and INLINEFORM3 are two semantically equivalent or similar LLTs (i.e., LLTs with the same PT) retrieved by the human and the automatic encoding, respectively; we use bold type to denote terms that have been recognized by MagiCoder but that have not been encoded by the reviewer; we use italic type in D1, D2, D3 to denote text recognized only by MagiCoder. For example, in description D3, “cefalea” (in English, “headache”) is retrieved and encoded both by the human reviewer and MagiCoder; in description D2, ADR “febbre” (in English, “fever') has been encoded with the term itself by the algorithm, whereas the reviewer encoded it with its synonym “piressia”; 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 INLINEFORM0), but it has been pruned from the list of the winning terms by the ordered-phrase heuristic criterium.

Discussion

We discuss here some interesting points we met developing MagiCoder. We explain the choices we made and consider some open questions.

Stemming and performance of the NLP software

Stemming is a useful tool for natural language processing and text searching and classification. The extraction of the stemmed form of a word is a non-trivial operation, and algorithms for stemming are very efficient. In particular, stemming for Italian language is extremely critic: this is due to the complexity of language and the number of linguistic variations and exceptions.

For the first implementation of MagiCoder as VigiFarmaco plug-in, we used a robust implementation of the Italian stemming procedure. The procedure takes into account subtle properties of the language; in addition of the simple recognition of words up to plurals and genres, it is able, in the majority of cases, to recognize an adjectival form of a noun by extracting the same syntactical root.

Despite the efficiency of this auxiliary algorithm, we noticed that the recognition of some MedDRA terms have been 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, an open source text search engine library. This procedure is less refined 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. This induces a conservative approach and a uniform processing of the whole set of MedDRA words. This is unsatisfactory for a general problem of text processing, but it is fruitful in our setting. We repeated the MagiCoder testing both with the classical and the light stemmer: in the latter case, we measure a global enhancement of MagiCoder performance. Regarding common retrieved preferred terms, we reveal an average enhancement of about INLINEFORM0 : percentages for classes 1, 2, 3, 4 and 5 move from INLINEFORM1 , INLINEFORM2 , INLINEFORM3 , INLINEFORM4 , INLINEFORM5 , respectively, to values in Table TABREF58 . It is reasonable to think that a simple stemming algorithm maintains the recognition of words up to plurals and genres, but in most cases, the recognition up to noun or adjectival form is potentially lost. Notwithstanding, we claim that it is possible to reduce this disadvantage thanks to the embedding in the dictionary of a reasonable set of synonyms of LLTs (see Section SECREF66).

Synonyms

MagiCoder performs a pure syntactical recognition (up to stemming) of words in the narrative description: no semantical information is used in the current version of the algorithm. In written informal language, synonyms are frequently used. A natural evolution of our NLP software may be the addition of an Italian thesaurus dictionary. This would appear a trivial extension: 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 MagiCoder performances (both in terms of correctness and completeness): on average, common PT percentages decreases of 24%. The main reason is related to the nature of Italian language: synonymical groups include words related by figurative

meaning. For example, among the synonyms of the 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 the computational complexity. Thus, we claim that this is not the most suitable way to address the problem and the designing of an efficient strategy to solve this problem is not trivial.

We are developing a different solution, working side-by-side with the pharmacovigilance experts. The idea, vaguely inspired by the Consumer Health Vocabulary (recalled in Section SECREF2 and used in BIBREF16), is to collect a set of pseudo-LLTs, in order to enlarge the MedDRA official terminology and to generate a new ADR lexicon. This will be done on the basis of frequently retrieved locutions which are semantically equivalent to LLTs. A pseudo LLT will be regularly voted and sorted by MagiCoder and, if selected, the software will release 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 controls the complexity deterioration. 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” e “riduzione” (in English “increase”, “decrease”, and “reduction”, respectively) and their adjectival form. For each LLT containing one of these nouns (resp., adjectives) we generate an equivalent term taking into account the corresponding adjective (resp., noun).

This small set of synonyms induces a global improvement of MagiCoder performances on classes 4 and 5. For Class 4, both common retrieved PT percentage, precision and recall increase of INLINEFORM0 .

For Class 5, we observe some significative increment: common retrieved PT moves from INLINEFORM1 to INLINEFORM2 ; precision moves from INLINEFORM3 to INLINEFORM4 ; recall moves from INLINEFORM5 to INLINEFORM6 .

Also false negative and false positive rates suggest that the building of the MedDRA-thesaurus is a promising extension. False negatives move from INLINEFORM0 to INLINEFORM1 for Class 4 and from INLINEFORM2 to INLINEFORM3 for Class 5. False positive percentage decrease of INLINEFORM4 both for Class 4 and Class 5.

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

Connectives in the narrative descriptions

As previously said, in MagiCoder we do not take into account the structure of written sentences. In this sense, our procedure is radically different from those based on the so called part-of-speech (PoS) BIBREF29 , 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: 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. First, we evaluated the frequency of negation connectives in ADR reports: we considered the same sample exploited in Section SECREF52 , and we counted the occurrences of the words “non”

(Italian for “not”) and “senza” (Italian for “without”): we detected potential negations in 162 reports (i.e., only in the INLINEFORM0 of the total number, 4445). Even though negative sentences seem to be uncommon in ADR descriptions, the detection of negative forms is a short-term issue we plan to address. As a first step, we plan to recognize words that may represent negations and to signal them to the reviewer through the graphical UI. In this way, the software sends to the report reviewer an alert about the (possible) failure of the syntactical recognition.

On the selection of voted terms

As previously said, in order to provide an effective support to human revision work, it is necessary to provide only a small set of possible solutions. To this end, in the selection phase (described in Section UID28), we performed drastic cuts on voted LLTs. For example, only completely covered LLTs can contribute to the set of winning terms. This is clearly a restrictive threshold, that makes completely sense in a context where at most six solutions can be returned. In a less restrictive setting, one can relax the threshold above and try to understand how to filter more “promising” solutions among partially covered terms. In this perspective, we developed a further criterion, the Coverage Distribution, based on assumptions we made about the structure of (Italian) sentences. The following formula simply sums the indexes of the covered words for INLINEFORM0 : INLINEFORM1

If INLINEFORM0 is small, it means that words in the first positions of term INLINEFORM1 have been covered. We defined INLINEFORM2 to discriminate between possibly joint winning terms. Indeed, an Italian medical description of a pathology has frequently the following shape: name of the pathology+“location” or adjective. Intuitively, we privilege terms for which the recognized words are probably the ones describing the pathology. The addition of INLINEFORM3 (with the discard of condition INLINEFORM4 in the final selection) could improve the quality of the solution if a larger set of winning terms is admissible or in the case in which the complete ordered list of voted terms is returned.

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 “covers” the medical meaning of the free-text description. Differently from other BioNLP software proposed in literature, we developed an original text processing procedure. Preliminary results about MagiCoder performances are encouraging. Let us sketch here some ongoing and future work.

We are addressing the task to include ad hoc knowledges, as the MedDRA-thesaurus described in Section SECREF66 . We are also proving that MagiCoder is robust with respect to language (and dictionary) changes. The way the algorithm has been developed suggests that MagiCoder can be a valid tool also for narrative descriptions written in English. Indeed, the algorithm retrieves a set of words, which covers an LLT INLINEFORM0 , from a free-text description, only slightly considering the order between words or the structure of the sentence. This way, we avoid the problem of “specializing” MagiCoder for any given language. We plan to test MagiCoder on the English MedDRA and, moreover, we aim to test our procedure on different dictionaries (e.g., ICD-9 classification, WHO-ART, SNOMED CT). We are collecting several sources of manually annotated corpora, as potential testing platforms. Moreover, we plan to address the management of orthographical errors possibly contained in narrative ADR descriptions. We did not take into account this issue in the current version of MagiCoder. A solution could include an ad hoc (medical term-oriented) spell checker in VigiFarmaco, to point out to the user that she/he is doing some error in writing the current word in the free description field. This should drastically reduce users' orthographical errors without heavy side effects in MagiCoder development and performances. Finally, we aim to apply MagiCoder (and its refinements) to different sources for ADR detection, such as drug information leaflets and social media BIBREF16 , BIBREF30 .