Normalizing Spontaneous Reports into MedDRA: some Experiments with MagiCoder

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

***Abstract*—Text normalization into medical dictionaries is useful to support clinical task. A typical setting is Pharmacovigilance (PV). The manual detection of suspected adverse drug reactions (ADRs) in narrative reports is time consuming and Natural Language Processing (NLP) provides a concrete help to PV experts. In this paper we carry on experiments for testing performances of MagiCoder, an NLP application designed to extract MedDRA terms from narrative clinical text. Given a narrative description, MagiCoder proposes an automatic encoding. The pharmacologist reviews, (possibly) corrects, and then validates the solution. This drastically reduces the time needed for the validation of reports with respect to a completely manual encoding. In previous work we mainly tested MagiCoder performances on Italian written spontaneous reports. In this paper, we include some new features, change the experiment design, and carry on more tests about MagiCoder. Moreover, we do a change of language, moving to English documents. In particular, we tested MagiCoder on the CADEC dataset, a corpus of manually annotated posts about ADRs collected from social media.**

***Index Terms*—Natural language processing, Healthcare informatics, Pharmacovigilance, Adverse drug reactions, Term identification**

I. Introduction

NATURAL Language Processing plays a central role in healthcare informatics. Narrative descriptions of medical situations, e.g., clinical records, spontaneous reports and (in the last years) social media, are an irreplaceable source of information, as pointed out in [1]. This holds in particular for pharmacovigilance (PV), where the prompt detection of suspected adverse drug reactions (ADRs) is one of the main activities. Since premarketing studies about drug consumption effects are inconclusive (they are time bounded and cover only some categories of patients), data about suspected ADRs have to be collected once drugs are effectively marketed and used by citizens.

To improve the quality of data and simplify both reporting activities and the work of people responsible for PV, international authorities encourage the use of web-based formats and tools to report and manage ADRs (Directive 2010/84/EU, 2010; EU Regulation 1235/2010, 2010).

In most cases, European countries (e.g., Great Britain) offer to reporters a structured form to describe ADRs , typically a drop down menu or an autocompletion field. The reporter chooses, among the entries of the given clinical dictionary (in Italy, the MedDRA dictionary [2]), the most suitable (in her opinion) expression/locution for describing the disease she is reporting. This apparently good practice reveals at least two problems: (i) the description of an ADR by means of one of the thousand medical terms is a complex task, and (ii) the choice of the suitable term(s) from a given list or from an autocompletion field can influence the reporter and limit her expressiveness. For example, in MedDRA several terms about migraine (in Italian, "emicrania") exist (e.g., "migraine with aura", "basilar migraine", "vestibular migraine") and it may be confusing for a patient to choose among them.

As a consequence, the quality of the description would be also in this case undermined. Thus, describing ADRs by means of natural language sentences is simpler and preferable. In this way, for example, the patient has only to describe the characteristics of her migraine, without caring about what is the right medical term to use.

Once narrative data have been collected, the *text normalization* (or, equivalently, *text encoding*) problem occurs. Free text have to be mapped into medical dictionaries. Well-known examples of clinical dictionaries are MedDRA (Medical Dictionary for Regulatory Activities), a medical terminology used to encode adverse drug reaction information, ICD[1], the International Statistical Classification of Diseases and Related Health Problems, and the SNOMED-CT[2] (Systematized Nomenclature of Medicine - Clinical Terms), a multilingual clinical healthcare terminology used in clinical documentation and reporting.

Mapping text snippets to terms in a medical dictionary is time consuming. Moreover, in the last years spontaneous reports (mainly collected through the web) have grown exponentially[3], making the encoding task difficult and often unbearable for people responsible for PV. To support PV experts' normalization task we designed and implemented MagiCoder, an NLP software that, given a narrative description, automatically extracts codes from the MedDRA terminology. MagiCoder has been proposed in [3] and further developed and tested in [4]. In [5] we then proposed a modular, automatic evaluation of MagiCoder, testing some incremental versions of the software obtained by adding features to the basic version of the software.

With respect to [5], in this work we extensively describe some original extensions and discuss new experimental work. More precisely:

C. Combi and M. Zorzi are with the Dept. of Computer Science, University of Verona, Italy (e-mail: carlo.combi@univr.it; margherita.zorzi@univr.it).

G. Pozzani, E. Arzenton, and U. Moretti are with the Dept. of Diagnostics and Public Health, University of Verona, Italy (e-mail: gabriele.pozzani@univr.it; elena.arzenton@univr.it; ugo.moretti@univr.it).

[1] https://www.cdc.gov/nchs/icd/

[2] http://www.snomed.org/snomed-ct

[3] In Italy, the number of ADR reports passed from approximately 6000 in 2006 to more than 50000 in 2016 [3].

1) We further extend MagiCoder. We add to the features defined in [5] a new module called Ph − PhraseSplitting (Section III-A) to improve the performances.
2) We design some new experiments, evaluating the behaviour of single features on a basic version of the software. Features are added and tested individually. This is different from what we did in our prior work [5], where we observed the improvement of performance metrics along "incremental" versions of the software.
3) We test MagiCoder on the CADEC (CSIRO Adverse Drug Event Corpus) dataset, a corpus of annotated English ADRs posts from social media. Thus, we move from Italian to English narrative documents. Posts are completely different from spontaneous reports, and represent an interesting setting, since clinical information extraction from social media is one of the most intriguing trends in health NLP [6].

The paper is organised as follows: in Section II some background notions are reported about the MedDRA dictionary; in Section III we describe MagiCoder main features and the different extensions of the software. Section IV describes the performances of MagiCoder on a dataset of Italian spontaneous reports, while in Section V we report the performances of MagiCoder on the CADEC dataset. In Section VI we discuss related work. Finally, Section VII sketches some concluding remarks and future work.

II. Background

We now quickly consider the medical terminology we specifically used in this work. MedDRA is a medical terminology used to encode adverse event information associated with the use of biopharmaceuticals and other medical products [2]. MedDRA terms are organized into a hierarchy. The SOC (System Organ Classes) 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, HLT, and PT). Preferred Terms (PTs) are LLTs chosen to be the representative of a group of semantically equivalent terms. An example of the hierarchy is: the reaction *Itch* (LLT), described starting from *Skin disorders* (SOC), through *Epidermal conditions* (HLGT), *Pruritus NEC* (HLT), and *Pruritus* (PT).

III. The Software MagiCoder: Basic Features and Modular Extensions

MagiCoder is an NLP tool developed for Italian pharmacovigilance, to support the detection and the classification of free text descriptions of adverse drug reactions. MagiCoder normalizes narrative descriptions about ADRs into the MedDRA dictionary and offers to the PV experts a possible encoding. The responsible experts can accept the encoding as it is or modify the automatic solution.

MagiCoder is not based on Part-of-Speech (PoS) tagging methods, as the majority of NLP software tools. MagiCoder exploits a simpler idea. A single scan of the original description is enough to retrieve information. From an abstract point of view, we try to recognize, in the narrative description, *single words* (*tokens*) belonging to LLT terms, where recognised words do not necessarily occupy consecutive positions in the text. In this way, we try to *"rebuild"* MedDRA terms, taking into account the fact that in a description the reporter can permute or omit words. We call tokens in the text and in LLTs involved in the perfect matching *voters* and *voted*, respectively. This terminology comes from the fact that, in some sense, MagiCoder uses a "voting" procedure. Tokens "vote" (i.e., recognize) LLTs matching their words. Voters and voted tokens do not necessarily occupy consecutive positions in the text or in the LLT, and their order does not matter. After the linear scan of the text, each MedDRA term including at least a token in the description (voted terms) is marked with additional information, such as how many of its tokens have been recognized in the report. Thus, voted terms are ordered by some criteria, and only the subset of them that passes some given thresholds are finally proposed as the automatic solution. MagiCoder does not involve computationally expensive tasks, such as subroutines for permutations and combinations of words. We quickly summarize here the main ideas of MagiCoder [3].

1) *Preprocessing of the original text*. We perform standard operations such as tokenization, elimination of irrelevant words (*stop-words*), stemming.
2) *Word-by-word linear scan* of the description and "voting task". A word "votes" LLT terms it belongs to. For each term voted by one or more words, we store some information about the retrieved syntactical matching, such as relative position of voters in the text and in the voted LLT.
3) *Weights calculation*. Recognised terms (terms voted by at least one word of the text) are weighted according to information about the syntactical matching. Main weights (ordered by priority) are: coverage, i.e., the estimation of the percentage of the LLT recognised in the narrative description; kind of coverage, i.e., the recognition of a word in its non stemmed form receives a better evaluation; density, i.e., the "quality" of the coverage.
4) *Sorting of voted terms*. Voted LLTs are (multi-valued) sorted by their weights.
5) *Winning terms release*. A solution, a set of winning terms, is released. The main idea is that the procedure scans the list of ordered terms and try to "cover", as best as possible, the textual description.

Fig. 1 summarizes MagiCoder main steps.

The use of suitable data structures such as hash tables is central for MagiCoder efficiency. The MagiCoder abstract computational complexity is $O(mk) + O(nm)$ time units, where n is the input size (in terms of tokens), m is the size of the dictionary (the number of terms), and k is the length (in terms of tokens) of the longest term in the dictionary. In particular, the setting up phase (required only once when the tool starts, in order to build auxiliary hash tables storing words of MedDRA) requires $O(mk)$ time units (this is mandatory for testing perfect matching with words in the text). $O(nm)$ time units is the complexity of each encoding phase.

Thus, MagiCoder computational complexity is strongly related to the size of the considered dictionary.

Such worst-case complexity bounds are strongly pessimistic. Focusing on *effective* time performances of the pro-

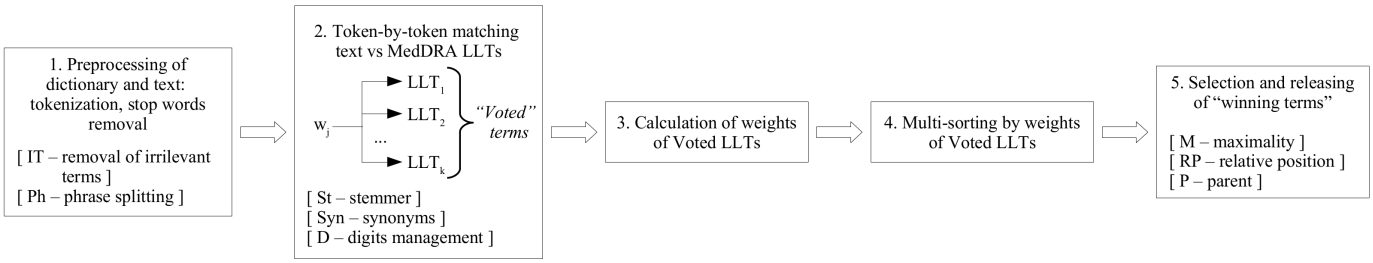

Fig. 1. High-level description of MagiCoder procedure

cedure, it is easy to observe that, during the computation, MagiCoder works on progressively smaller subsets of terms. This drastically reduces computation time of (theoretically) expensive subroutines. It is easy to prove that, working with the MedDRA dictionary, the effective complexity can be rewritten as $O(n) + O(n'm')$. $n' \in O(n)$ represents the number of voters, it is n in the worst case, but it is drastically less than n in real computations (for example, $n' \simeq 1$ in the dataset we considered in Section IV). $m' \in O(m)$ represents the number of voted terms, it is m in worst case, but, as suggested by our experiments, it is very small in real applications. In [5] further details are provided. We report some statistics about MagiCoder execution time. Tests have been performed implementing MagiCoder in Ruby 2.3.0 (without a multi-processor/multi-thread support) on a desktop PC equipped with a CPU IntelCore i5-4570 @ 3.2 GHz and running Ubuntu Linux 14.04.5. On average, on the dataset of narrative descriptions described in Section IV-A, MagiCoder required 2.77ms for encoding the reports of length up to 20 characters, 5.61ms for reports of length from 21 to 40 characters, 12.43ms for reports of length from 41 to 100 characters, 25.72ms for reports of length from 101 to 255 characters, and 37.57ms for reports longer than 255 characters, respectively.

A. MagiCoder Extensions

We describe now the modules of MagiCoder we designed and developed. We also provide some examples illustrating each feature. Modules can be added to MagiCoder "skeleton" (Fig. 1) one-by-one, simultaneously or incrementally. We call "configuration" a particular combination of features.

In [5] we considered the MagiCoder skeleton as basic version. Here we consider as the basic version a particular configuration of the software including some default modules. We first describe modules we included in the basic version.

Maximality criterion (M). This is a module implementing a heuristic criterion, based on the idea that the best solution is the more detailed one. Roughly speaking, we remove from the solution each term that can be considered "subset" of another one (i.e., there exists another term, which extends it with further words). For example, *hepatitis fulminant* specializes and refines the ADR *hepatitis*, and then MagiCoder chooses the first one.

Relative position criterion (RP). Module RP introduces the order of words in the narrative description as a selection discriminating factor. From the set of selected LLTs, it removes 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. The criterion is applied only to LLTs which voters have voted also for other terms. For example, consider the description "*erythema and abdominal pain*". MagiCoder returns *abdominal pain* and *erythema* but does not return *abdominal erythema*.

Parent criterion (P). During the release of the solutions, this criterion removes an LLT from the list of the winning terms if another term with the same PT already belongs to the solution. For example, only one term among "fever" and "pyrexia" will be returned.

Digits management (D). Digits (e.g., codes and numeral adjective) frequently occur in ADR descriptions. We try to improve MagiCoder behavior by adding the capability to deal with digits (digits are treated as irrelevant information by default). This feature has been used, for example, in conjunction with synonyms (see later in this section) for detecting hyperpyrexia described as "fever over 40". In this case the reported number is crucial because, conversely, "fever of 39" has to be encoded as "pyrexia". Similar examples are available for several other lab tests that may be encoded in a different MedDRA term, depending on the reported result.

The extensions described above are included in MagiCoder basic version, hereinafter named B.

We now describe modules we will focus on in the experiments, adding them to version B and observing their effects on performances.

Exclusion of irrelevant dictionary terms (IT). Some LLTs are irrelevant for the purpose of identifying ADRs (e.g., LLTs describing social conditions). Experts of the domain suggested a list of uninformative terms. This module instructs MagiCoder to disregard all these terms. Examples of irrelevant terms are those related to the patient's social conditions and circumstances (e.g., "stress at work", "alcoholic relative"), which may help physicians to understand the patient but do not represent drug reactions. Similarly, also terms representing lab test (e.g., "Citric acid urine") have to be disregarded.

Stemmer (St). This extension introduces two different versions of stemming procedure both for Italian and English. For both languages the first stemmer (called pSt) is based on Porter's algorithm [7]. Based on a list of predefined suffixes, it removes the last part of words. For example, the Porter's stem of "psychiatric" is "psychiatr". The second stemmer (called lSt) for Italian language is based on the algorithm described in [8]. It can be considered as a "light" stemmer

because it breaks off only the last characters of the last syllable. This induces a conservative approach and a uniform processing of the whole set of MedDRA words. This should be unsatisfactory for a general problem of text processing, but it is fruitful in our setting. For example, the light stem of "psichiatrico" (in English, psychiatric) is "psichiatric". On the other hand, for English language lSt is based on the Lancaster algorithm[4]. Lancaster procedure is definitively more aggressive than Porter's method and returns very short stems. For example, the Lancaster's stem of "psychiatric" is "psychy".

Addition of synonyms (Syn). This module equips MagiCoder with synonyms of some LLTs, i.e., locutions semantically equivalent to the official ones.

For Italian reports, we considered about 4500 synonyms. The additional lexicon has been generated in different ways. A subset of terms has been automatically generated by considering words that frequently occur in MedDRA, either as nouns or as adjectival forms. New LLTs have been created by replacing nouns (resp., adjectives) with the corresponding adjective (resp., noun). For example, from "acido folico aumentato" (in English, folic acid increased) we obtained "aumento acido folico" (in English, increase folic acid). Other terms can be easily obtained by replacing words in LLTs with frequently used abbreviations or acronyms or vice versa. For example, from "cataratta destra" (in English, right cataract) the synonym "cataratta dx" is generated. Another set of synonyms has been automatically generated with statistical methods for linguistics from a corpus of about 250,000 narrative ADR reports [9]. This set has been successively validated by an expert of the domain, and finally added to the lexicon. The same methods have been also applied for generating synonyms from the English corpus. We will further discuss it in Section V. We note here that all synonyms have been validated by domain experts, together with the methods used to generate them.

Phrase Splitting (Ph). We try to introduce some hypotheses about the structure of the text, exploiting punctuation marks (dots, semicolons, colons, commas) and brackets. We aim at managing long texts, breaking them into shorter descriptions. In the next sections, we discuss some tests, in particular observing the effects of treating comma in different ways. In a first setting, called Ph w/o comma, we ignore commas and we break phrases only according to other marks. In other settings, dubbed as Ph comma k, we treat comma as a "soft" connective. If two words are separated by a comma, we consider them as being k (a parameter) tokens apart. Since nearness of words to each other is used for the terms selection, this has an impact on MagiCoder solution release. We consider both $k = 1$ and $k = 2$. Finally, we treat commas as other punctuaction marks, and we break text according to the presence of a comma (we dub it as Ph comma inf, since it is equivalent to set k to infinite).

Consider for example the ADR description "fever, bronchitis chronic". Without taking into account the comma, the adjective "chronic" would be associated both to "fever" and "bronchitis". On the other hand, with Ph comma inf the description is broken in two distinct phrases and "chronic" will be associated only to "bronchitis".

These last four features may be independently added to the basic version of MagiCoder.

In the following sections we measure the increase (or decrease) of some standard metrics adding modules IT, St, Syn, and Ph one-by-one to the basic version B, that includes, as previously specified, M, RP, PT, and D.

Differently from experiments performed in [5], tests in Section IV and V do not reproduce the evolution of the software along different versions, where a feature is added to a "stack" of previously tested modules in order to incrementally improve performances. Instead, here we observe the effects related to "turning on" a single module on the basic version B. This allows us to study the impact and importance of any single feature on the MagiCoder performances. Moreover, after testing modules one-by-one, we provide an "overall experiment", testing the best combination of all modules. Where different options are available, as in the case of stemmer, we will choose the best one in terms of performances.

In Section IV and V we we discuss the tests we performed on MagiCoder on Italian and English texts, respectively.

IV. Experiments on Italian Narrative Reports

Performances of MagiCoder are measured by two well-known metrics, namely *precision* (P, also called positive predictive value) and *recall* (R, also called sensitivity). Both are defined in terms of *false positive* (FP) and *false negative* (FN) errors, the two main kinds of error in binary classification [10]. In our case, an FP error corresponds to the retrieval of a "wrong" LLT, i.e., a term that does not correctly encode the textual description. An FN error corresponds to the failure in the recognition of a "good" LLT, i.e., a term that effectively encodes (a part of) the narrative description and that would be selected by a human expert. As dual notions of false positive and false negative, one can define *correct* results, i.e., *true positives* (TP), i.e., correctly returned LLTs and *true negative* (TN), i.e., LLTs which, correctly, are not proposed 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. Precision and recall are formally defined as $\mathsf{P} = \frac{TP}{TP+FP}$ and $\mathsf{R} = \frac{TP}{TP+FN}$, respectively.

Informally, precision measures the portion of the returned results (i.e., LLTs) that is actually correct, while recall measures the portion of correct results (w.r.t. the gold reference standard) that is actually returned. Precision and recall will be used both for experiments on Italian texts and English texts (Section V).

A. Experiment Design

To test MagiCoder on Italian, we exploited a corpus of formerly manually revised and annotated ADR reports.

The dataset includes 4500 ADR reports concerning Regione Veneto (Italy) from the data warehouse Vigisegn [11], developed for the Italian pharmacovigilance activities. We divided reports in five classes according to their length. TABLE I

[4]https://github.com/words/lancaster-stemmer

summarizes, for each class, the range of lengths considered and the number of reports. Classes have been selected to reflect in the dataset the actual distribution of text lengths in the whole database.

Reports have been manually annotated by PV experts. These manual revisions represent our *reference standard*. To evaluate MagiCoder performances, we *automatically* compared (by a benchmark) the output provided by each MagiCoder version with the manual encoding of the ADR reports from the dataset. For each report, the test verifies which terms belong to both or only one of manual and automatic solutions, and then calculates errors and performance measures. Results are discussed in Section IV-B. It is worth noting that this test is *quantitative*. It simply estimates how much, for each report, MagiCoder behavior is similar to the manual one. Conversely, it is not able to fairly test *redundancy errors*. Human experts make some encoding choices in order to avoid repetitions. A term returned by MagiCoder that has not been selected by the expert because redundant should not be a false positive result. For example, consider the following Vigisegn report (here translated in English): "anaphylactic shock (hypotension) 1 hour after taking the drug.". Since "hypotension" is considered a well-known symptom of "anaphylactic shock", a PV expert should avoid the encoding of "hypotension", releasing only the main pathology "anaphylactic shock". Instead, MagiCoder returns both terms, and this solution is clearly correct. This suggests that we are underestimating MagiCoder performances,. as also shown in [4]. This kind of semantic information could be extracted from suitable MedDRA ontologies. We plan to equip MagiCoder with further knowledge bases and this is an interesting future work.

In [5] we performed a careful analysis of the recall and precision trends w.r.t. different versions of MagiCoder. We started by testing the skeleton version in Fig. 1 (MagiCoder without any heuristic or knowledge base) and then we progressively and cumulatively added modules described in Section III-A, with the exception of Ph. In the present study we choose a different perspective. We test each module at a time, measuring its improvement of the performances w.r.t. the basic version. We finally also test a "best configuration" of MagiCoder, that includes all features with their best setting.

In the following sections, P and R are always calculated report by report. When we detail metrics concerning a specific class, we average P and R values of reports in that class, respectively. On the other hand, the overall performances, which do not take into account description lengths, are computed by averaging P and R of all documents in the dataset.

B. Results

We computed results considering terms at PT level, both for the human and the automatic encoding. By moving to PT level, instead of using the MedDRA LLT level, we group together terms that represent the same medical concept (i.e., the same adverse reaction, as for example "fever" and "pyrexia"). In this way, we try to not consider as an error a case in which MagiCoder and the human expert use two different LLTs for representing the same adverse event. The use of the LLT level for reporting purposes and the PT level for analysis is suggested also by MedDRA [2].

TABLE I
BASIC VERSION B: DETAILED PERFORMANCES ON VIGISEGN DATASET

Class	Length (chars)	No of reports	avg PTs R	avg PTs P
0	from 0 to 20	459	87.00%	89.70%
1	from 21 to 40	1012	70.22%	75.55%
2	from 41 to 100	1998	60.04%	63.57%
3	from 101 to 255	1020	57.30%	53.80%
4	more than 255	11	50.74%	48.33%
Overall		4500	64.01%	66.15%

TABLE II
MAGICODER EXTENSIONS: OVERALL PERFORMANCES ON VIGISEGN DATASET

Module	avg PTs R	avg PTs P	R var. w.r.t. B	P var. w.r.t. B
B	64.01%	66.15%		
IT	65.66%	71.36%	1.65%	5.20%
Syn	66.22%	66.85%	2.22%	0.70%
pSt	65.77%	63.54 %	1.76%	-2.61%
lSt	65.68%	64.76%	1.67%	-1.39%
Ph w/o comma	63.94%	66.27%	-0.08%	0.11%
Ph comma 1	63.91%	66.31%	-0.09%	0.16%
Ph comma 2	63.96%	66.40%	-0.05%	0.25%
Ph comma inf	63.94%	66.42%	-0.07%	0.27%

TABLE I summarizes performances for the basic version B on the Vigisegn dataset. In this case we also detail recall and precision of each class and the overall performances without taking into account descriptions length.

TABLE II depicts the trends of recall and precision when we turn on, one-by-one, each MagiCoder extension. Even though the table shows only the overall performances, we discuss here some further details about the performances on specific classes (not shown in the table). Module IT clearly increases precision. Focusing on interesting classes 2 and 4, we have an improvement of 4.94% and 10.19%, respectively. The most evident effect of the addition of synonyms (globally increasing R of 2.22%) is again for class 4, where the improvement of R reaches 6.96%. Stemming algorithms clearly affect recall, inducing at the same time a natural worsening on P. It is evident that Porter's algorithm globally behaves worse, since precision decreases of 2.61% (versus the 1.39% of the light stemmer). Thus, light stemmer is preferable in our setting. Module Ph, which implements the division of sentences in smaller ones, has little effect on recall and precision. Once again, quite interesting results come from longer descriptions. On class 4, we have an increase in precision of 0.56% both for Ph comma 2 and Ph comma inf. This also suggests that it is better to break phrases according to commas. This is reasonably related to how reports are written.

Finally, TABLE III offers, detailing each class of length, results for the "best" configuration of MagiCoder, i.e., in this case, the configuration obtained by adding to B all modules, choosing lSt as stemmer algorithm and Ph comma 2 as division criterion. Overall increasing of recall and precision w.r.t. B are 4.84% and 3.41%, respectively. Focusing on long reports, the enhancement is more evident, with an improvement of 6.56% of recall and 12.43% of precision. Globally, in this experiment recall reaches 68.85% and precision reaches 69.96%. On reports up to 100 character (a reasonable bound to propose to

TABLE III
BEST CONFIGURATION: PERFORMANCES ON VIGISEGN DATASET

Class	PTs R	PTs P	R var. w.r.t B	P var. w.r.t B
0	88.61%	90.22%	1.61%	0.52%
1	74.79%	77.21%	4.47%	1.66%
2	64.44%	66.65%	4.40%	3.08%
3	63.68%	61.13%	6.38%	7.33%
4	57.30%	60.76%	6.56%	12.43%
Overall	68.85%	69.96%	4.84%	3.81%

TABLE IV
BASIC VERSION B: PERFORMANCES ON THE CADEC DATASET

Class	Length (chars)	No of posts	avg PTs R	avg PTs P
0	from 0 to 150	170	64.44%	70.10%
1	from 151 to 300	226	57.67%	61.10%
2	from 301 to 450	224	53.57%	55.95%
3	from 451 to 700	224	54.65%	53.68%
4	more than 700	255	53.36%	44.86%
Overall		1099	56.14%	55.91%

reporter), we measure 76% for recall and 78% for precision.

V. EXPERIMENTS ON THE CADEC CORPUS

MagiCoder has been developed to be robust w.r.t. the change of the language and the dictionary. First, we had some initial evidence about the fact that MagiCoder well performs also on English descriptions. To this regard, we processed a small subset of summaries of product characteristics and compared the automatic solution with the encoding of a PV expert. Here, we discuss more in-depth experiments.

A. Experiment Design

In this experiment we exploit CADEC (CSIRO Adverse Drug Event Corpus) [12]. CADEC is an annotated corpus of medical forum posts on patient-reported ADRs. The corpus is sourced from *posts from social media*, and contains text that is largely written in colloquial language. CADEC jargon is interesting, since it often deviates from formal English grammar. Authors annotated descriptions linking concepts such as drugs, adverse effects, symptoms, to their corresponding concepts in SNOMED-CT and MedDRA vocabularies.

The corpus includes 1099 reports, divided according to their lengths as described in TABLE IV. In CADEC, length distribution is radically different w.r.t. the Italian dataset. In this case long reports cover the larger part of the dataset. This is interesting for our experiment, since it permits us to test MagiCoder on very long descriptions.

On the CADEC corpus, we perform the same runs of MagiCoder described in the previous section. We start from B and we turn on each module one-by-one. Clearly, we change stemming procedure. In this case we adopt the Porter's stemmer for English (pSt) and the Lancaster algorithm (lSt). Moreover, we also change the set of synonyms, according to the language change. Notice that the set of irrelevant terms (module IT) is the same for all languages, since it depends on MedDRA dictionary.

For this experiment we created a set of *ad hoc* synonyms for the CADEC corpus. The protocol, that involves a PV expert, is semi-automatic and is described in [9]. We quickly recall here the main ideas. Computational linguistics research proved that statistics plays a major role in the problem of *lexicon generation* [13]. In particular, *co-occurrence based* techniques perform well in the tricky task of generating technical/scientific terminology. Roughly speaking, a co-occurrence represents a pair of words that frequently appear together in the same documents of a given dataset. Statistical methods for linguistics tell us that (semantically) related words often co-occur. As a consequence, a co-occurrence pair with a high frequency has a great probability to represent a meaningful association. In our setting, after a suitable choice of the dataset and of the statistical scores, we tried to automatically generate *new locutions* equivalent to official LLTs. As a second step, a PV expert checked and eventually approved the new terminology, and finally linked it to its official synonym. For example, the computation of co-occurrence pair done for our test yields, among the results, the pair $\langle cramps, hunger \rangle$. A PV expert checked the pair and linked it to the official English LLT "*abdominal cramps*". Notice that the completely manual generation of synonyms during the everyday encoding work is a good modus operandi, but it is not trivial and it requires a huge amount of work by an expert of the domain. Moreover, for human experts it is more difficult to analyze and filter locutions by frequency (a meaningful medical synonym is useless if, de facto, it is rarely used in free-text reporting). Statistical techniques drastically reduce the time and generates only significant synonyms.

In the present work we computed co-occurrences, i.e., candidate synonyms, using the CADEC dataset. After the PV expert validation (a work that lasted about two hours), we generated 90 English locutions equivalent to MedDRA LLTs.

B. Results

TABLE IV summarizes performances for the basic version B on CADEC dataset. As for the Vigisegn dataset, we also detail results about each class.

It is worth nothing that CADEC is a very challenging dataset, since it includes, as previously said, very long and colloquially written description. Thus, it is not surprising that, on B, performances seem to be worse than for Italian. It is important to say that a direct comparison does not make sense, and not only for the difference in length distributions. CADEC is a set of posts, i.e., documents completely different from spontaneous reports in Vigisegn. Anyway, it is interesting to observe that, on CADEC corpus, modular features plugged in the basic version have a more evident (positive) impact on performances. Average results about MagiCoder performances according to modular extensions are reported in TABLE V.
As for Table II, we only report overall results but we discuss, when interesting, also performances on specific classes. As for Italian language, module IT improves precision, with an overall speedup of 9.64%. In particular, for very long descriptions in class 4 we observe an improving of 12.46%. The addition of synonyms increases recall of 2.17%. Considering that we added "only" 90 synonyms, this is a good result. This considerable effect against a relatively poor synonym knowledge base is due to the ad hoc generation of co-occurrence pairs

TABLE V
MAGICODER EXTENSIONS: OVERALL PERFORMANCES ON CADEC DATASET

Module	avg PTs R	avg PTs P	R var. w.r.t. B	P var. w.r.t. B
B	56.14%	55.90%		
IT	56.93%	65.54%	0.79%	9.64%
Syn	58.31%	56.03%	2.17%	0.12%
pSt	65.50%	53.11%	9.36%	-2.80%
lSt	66.03%	51.00%	9.89%	-4.90%
Ph w/o comma	56.24%	56.59%	0.10%	0.69%
Ph comma 1	56.06%	56.80%	-0.07%	0.90%
Ph comma 2	55.83%	57.16%	-0.3%	1.25%
Ph comma inf	55.90%	57.61%	-0.24%	1.71%

TABLE VI
BEST CONFIGURATION: PERFORMANCES ON CADEC DATASET

Class	PTs R	PTs P	R var. w.r.t B	P var. w.r.t B
0	78.83%	78.63%	14.39%	8.52%
1	68.99%	67.10%	11.32%	6.00%
2	65.15%	58.83%	11.58%	2.88%
3	66.17%	59.92%	11.52%	5.24%
4	63.11%	51.19%	9.75%	6.33%
Overall	67.70%	61.90%	11.56%	6.00%

described above. Focusing on stemming algorithms, Lancaster algorithm behaves worse, since precision decreases of 4.90%, versus the 2.80% of Porter's stemmer. The latter is thus preferable. For Ph module and its versions, considerations done for the Italian experiment still remain valid. On the CADEC dataset the effect on precision is definitively more visible. This is due to the fact that the average length of CADEC posts is longer than the average length of Vigisegn reports. Globally, the best solution is again to consider comma as an effective division (configurations Ph comma 2 and Ph comma inf).

In TABLE VI we report results for the best configuration of MagiCoder, i.e., for the CADEC case, the configuration obtained by adding to B all modules, choosing pSt as stemmer algorithm and Ph comma inf as division criterion. Overall increasing of recall and precision w.r.t. B are 11.56% and 6.00% respectively. Globally, in this experiment MagiCoder recall reaches 67.70% and precision reaches 61.90%. On posts up to 100 characters, we measure 70.99% for recall and 68.19% for precision.

VI. RELATED WORK

Automatic detection of ADRs from text (e.g., electronic health records, spontaneous reports, social media posts) has received an increasing interest in pharmacovigilance research. Wang et al. [14] conduct an experiment by collecting narrative discharge summaries from the Clinical Information System at New York Presbyterian Hospital through the NLP system MedLEE [15]. The authors identify clinical events and pathologies, which could be potential adverse drug events, reaching an overall recall and precision of 75% and 31% (for known ADRs), respectively. Coffman et al. [16] develop an algorithm in order to help coders in the subtle task of auto-assigning International Code of Disease (ICD-9 codes) to clinical narrative descriptions. ICD-9 is also used as controlled vocabulary by Ribeiro et al. [17], who evaluate the retrieval performance of a tool (based on information retrieval techniques without the need of supervision and of training data) that automatically categorises medical documents. Performances are evaluated on a data set of about 20k documents, and authors reveal the algorithm attains levels of average precision around the 70-80% range. Shultz et al. [18] introduce MEDSYNDIKATE, a natural language processor that couples grammatical knowledge, semantic knowledge, and conceptual (ontological) knowledge, and automatically acquires medical information from pathology-oriented clinical reports.

Healtcare social media represent an emerging reality in bio NLP. Yang et al. [19] propose to use association mining and proportional reporting ratio (PRR, a well-known pharmacovigilance statistical index) to mine the associations between drugs and adverse reactions from the user-contributed content in social media. Social media are also central in the work by Metke-Jimenez and Karimi, where authors use the CADEC dataset (the same we used Section V) to compare different algorithms for ADRs extraction in medical social media [20], [21]. The global task is divided into two sub-problems, i.e., concept identification followed by normalization into controlled vocabularies. Singular sub-tasks are first tested separately and then globally evaluated. Authors consider MetaMap[5] as baseline and evaluate both dictionary- and machine learning-based approaches. They obtain the best performances by using Conditional Random Fields (CRF) [22]. In particular, with CRF, they measured a precision of 64.4% and a recall of 56.5% for the concept identification task. They measured also an effectiveness (a measure based both on binary classification and ontological information) of 37.76% for the normalization task (CRF is in this case used in a mixed approach together with the semantic knowledge base Ontoserver[6]). Finally, CRF gained a precision of 77.1% and a recall of 37.6% for the full task.

Also machine learning based approaches are widely exploited in ADRs extraction. Several tools analyzed in [20] fall in this category and they are indubitably powerful. Since our work follows a radically different approach, we only recall here the method developed by Attardi et al [23], particulary relevant in our setting since it is oriented to Italian language. They use machine learning techniques to address knowledge extraction (e.g., relevant entities such as symptoms, diseases, and treatments) from clinical records written by Italian physicians. Experiments are performed extending semi-automatically generated corpora and defining ad hoc new sets of annotated documents. We refer the reader to literature for other bioNLP tools based on machine learning (in particular [24], [25] that address ADRs extraction from social media).

Other contributions about pharmacovigilance and the role of NLP in clinical decision support are [1], [26]–[32].

VII. DISCUSSION AND CONCLUSIONS

In this paper we carried on some experiments about MagiCoder, an NLP software developed to help PV experts in encoding narrative text into MedDRA terms. Everyday practice

[5]https://metamap.nlm.nih.gov/
[6]http://ontoserver.csiro.au:8080/

confirms that MagiCoder effectively supports experts' work, reducing time and improving the accuracy of the encoding.

Results of Section IV and V show that MagiCoder is efficient, also considering that the automatic comparison between automatic and human solutions we conducted underrates performances. Results on English language (CADEC dataset) are encouraging also by considering the recent results discussed in [20], [21] with respect to ADRs extraction from medical forums. Notwithstanding, MagiCoder's efficiency can be further improved, by refining heuristics and enriching the knowledge bases of the algorithm.

Since MagiCoder's performances are considered satisfactory by Italian PV experts using the software in their daily tasks, from now on we plan to develop the tool and test mainly focusing on the English language and on the CADEC corpus we used in this work. CADEC is particularly interesting for at least two reasons. It offers a tricky jargon and CADEC documents are based on posts from social media, which analysis is one of the most intriguing trends in biomedical NLP we aim to address as a short time goal. The experiments proposed in this paper can be extended in several ways. In particular, we aim to create a more refined experiment involving PV experts, to overcome limits induced by the automatic comparison between algorithmic and human solution, to obtain a more fair measure of performances. Moreover, we plan to further develop knowledge bases for English language. Co-occurrence based techniques [9] used for English dataset in Section V offer further possible improvements. These methods perform better on very large number of documents. We aim to generate synonyms for the CADEC corpus taking into account other significant external datasets. Moreover, differently from languages as Italian, the Unified Medical Language System (UMLS)[7] should represent another fruitful source for additional ADR lexicon.

REFERENCES

[1] S. Meystre and P. J. Haug, "Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation," *J Biomed Inform*, vol. 39, no. 6, pp. 589 – 599, 2006.

[2] ICH, "MedDRA data retrieval and presentation: points to consider," 2017.

[3] M. Zorzi, C. Combi, R. Lora, M. Pagliarini, and U. Moretti, "Automagically encoding adverse drug reactions in MedDRA," in *IEEE Int. Conference on Healthcare Informatics, ICHI 2015*, 2015, pp. 90–99.

[4] C. Combi, M. Zorzi, G. Pozzani, U. Moretti, and E. Arzenton, "From narrative descriptions to MedDRA: automagically encoding adverse drug reactions," *Journal of Biomedical Informatics*, 2018, in press.

[5] M. Zorzi, C. Combi, G. Pozzani, and U. Moretti, "Mapping free text into MedDRA by natural language processing: A modular approach in designing and evaluating software extensions," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2017, Boston, MA, USA, August 20-23, 2017*, 2017, pp. 27–35.

[6] A. Sarker and et al., "Utilizing social media data for pharmacovigilance: A review," *Journal of Biomedical Informatics*, vol. 54, 2015.

[7] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[8] J. Savoy, "Report on CLEF-2001 experiments," Université de Neuchâtel, Switzerland, Tech. Rep., 2001.

[9] M. Zorzi, C. Combi, G. Pozzani, E. Arzenton, and 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*, ser. Lecture Notes in Computer Science, vol. 10259. Vienna, Austria: Springer, June 21-24 2017, pp. 215–220.

[10] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, USA: Cambridge University Press, 2008.

[11] R. Lora, A. Sabaini, C. Combi, and U. Moretti, "Designing the reconciled schema for a pharmacovigilance data warehouse through a temporally-enhanced ER model," in *2012 Int. Workshop on Smart Health and Wellbeing*, ser. SHB 2012. ACM, 2012, pp. 17–24.

[12] S. Karimi, A. Metke-Jimenez, M. Kemp, and C. Wang, "CADEC: A corpus of adverse drug event annotations," *Journal of Biomedical Informatics*, vol. 55, pp. 73–81, 2015.

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

[14] X. Wang, G. Hripcsak, M. Markatou, and C. Friedman, "Active computerized pharmacovigilance using natural language processing, statistics, and electronic health records: A feasibility study," *J Am Med Inform Assoc.*, vol. 16, no. 3, pp. 328–337, 2009.

[15] C. Friedman, "Discovering novel adverse drug events using natural language processing and mining of the electronic health record," in *12th Conference on Artificial Intelligence in Medicine, AIME 2009*, ser. LNCS, vol. 5651. Springer, 2009, pp. 1–5.

[16] A. Coffman and N. Wharton, "Clinical natural language processing: Auto-assigning ICD-9 codes," in *Overview of the Computational Medicine Center's 2007 Medical Natural Language Processing Challenge*, 2007.

[17] B. Ribeiro-Neto, A. H. Laender, and L. R. de Lima, "An experimental study in automatically categorizing medical documents," *Journal of the American Society for Information Science and Technology*, vol. 52, no. 5, pp. 391–401, 2001.

[18] U. Hahn, M. Romacker, and S. Schulz, "MEDSYNDIKATE natural language system for the extraction of medical information from findings reports," *International Journal of Medical Informatics*, vol. 67, no. 1, pp. 63 – 74, 2002.

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

[20] A. Metke-Jimenez and S. Karimi, "Concept extraction to identify adverse drug reactions in medical forums: A comparison of algorithms," *CoRR*, vol. abs/1504.06936, 2015. [Online]. Available: http://arxiv.org/abs/1504.06936

[21] ——, "Concept identification and normalisation for adverse drug event discovery in medical forums," in *Proceedings of the First International Workshop on Biomedical Data Integration and Discovery (BMDID 2016)*, Kobe, Japan, October 2016.

[22] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Found. Trends Mach. Learn.*, vol. 4, no. 4, pp. 267–373, 2012.

[23] G. Attardi, V. Cozza, and D. Sartiano, "Annotation and extraction of relations from italian medical records," in *Proceedings of the 6th Italian Information Retrieval Workshop, Cagliari, Italy, May 25-26.*, 2015.

[24] H. Gurulingappa, A. Mateen-Rajput, and L. Toldo, "Extraction of potential adverse drug events from medical case reports," *J Biomed Semantics*, vol. 3, no. 15, pp. 1–10, 2012.

[25] A. Sarker and G. Gonzalez, "Portable automatic text classification for adverse drug reaction detection via multi-corpus training," *J Biomed Inform*, vol. 53, pp. 196–207, 2015.

[26] R. Harpaz, H. S. Chase, and C. Friedman, "Mining multi-item drug adverse effect associations in spontaneous reporting systems," *BMC Bioinformatics*, vol. 11, no. S-9, 2010.

[27] N. Nissim and et al., "An active learning framework for efficient condition severity classification," in *Artificial Intelligence in Medicine (AIME'15)*, ser. LNCS. Springer, 2015, vol. 9105, pp. 13–24.

[28] A. Perez, K. Gojenola, A. Casillas, M. Oronoz, and A. D. de Ilarraza, "Computer aided classification of diagnostic terms in spanish," *Expert Systems with Applications*, vol. 42, no. 6, pp. 2949 – 2958, 2015.

[29] B. P. Ramesh, S. Belknap, Z. Li, N. Frid, D. West, and H. Yu, "Automatically recognizing medication and adverse event information from food and drug administration's adverse event reporting system narratives," *JMIR Medical Informatics*, vol. 2(1), 2014.

[30] L. Liu, N. Shorstein, L. Amsden, and L. Herrinton, "Natural language processing to ascertain two key variables from operative reports in ophthalmology." *Drug Safety*, vol. 26(4), pp. 378–385, 2014.

[31] D. D. Fushman, W. W. Chapman, and C. J. McDonald, "What can natural language processing do for clinical decision support?" *Journal of Biomedical Informatics*, vol. 42, no. 5, pp. 760 – 772, 2009.

[32] J. van der Zwaan, E. Tjong Kim Sang, and M. de Rijke, *An Experiment in Automatic Classification of Pathological Reports*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 207–216.

[7] https://www.nlm.nih.gov/research/umls/