Automate The Mapping Phase Of Drug Side Effects Using NLP Techniques

[1]Dr. D. V. Lalita Parameswari, [2]Y. Sowmya Reddy, [3]Dr. Ch. Mallikarjuna Rao

[1]Sr. Assistant professor,
Department of CSE
G. Narayanamma Institute of Technology Science (For Women), Hyderabad.
Email: lplalitap97@gmail.com

[2]Assistant professor,
Department of CSE
Anurag University,
Hyderabad.
Email: ysowmyacse@cvsr.ac.in

[3]Professor,
Department of CSE
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad 500090,
Telangana state, India
Email: chmksharma@yahoo.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: Drug safety is an important science to detect threats related to medication consumption. Drug monitoring is often referred to as drug defence. For example, medicine's side effects can be caused by interactions, excessive doses and violence. In addition, information must be collected and mapped to predefined terms to find models in which the unintended effects are induced. This mapping is today manually conducted by experts, which can be very time consuming and challenging. In this paper the aim is to automate the mapping phase of side effects using techniques for machine learning.The model was created using data of pre-existing mappings of literal side effect expressions. The last design used the pre-trained BERT language model and the latest findings were obtained inside the NLP. The final model was correct at 80.21 percent in the evaluation of the test range. Some wordings were found to be very difficult to define for our model, mainly because of uncertainty or lack of literal knowledge. Since a threshold was introduced that left the most difficult to identify wordings for manual mapping, which is very important to make mapping correctly. However, this method could still be improved because suggested terms were created from the model, which could be used as support for the manual mapping specialist.

Keywords: Machine Learning, NLP, BERT, MedDRA.

1. Introduction

Pharmacovigilance is defined by the World Health Organisation (WHO) as" the science and activities relating to the detection, assessment, understanding and prevention of adverse effects or any other possible drug-related problems." [1] Pharmacovigilance is important for identifying risks related to medicine intake caused by interactions, high dosage, misuse etc. where the detection of these risks can be crucial to ensure patient safety.

The Uppsala Monitoring Centre (UMC) is working alongside WHO as part of the WHO Programme for International Drug Monitoring (PIDM) for the common goal of "a systematic collection of information on serious adverse drug reactions during the development and particularly after medicines have been made available for public use".[2] One tool developed and maintained by UMC, for the purpose of drug safety, is VigiBase.[3] It is WHOs global database, containing millions of individual case safety reports (ICSRs) from countries all over the world that are part of PIDM. The ICSRs are reports containing suspected adverse drug reactions (ADRs) which are reported and collected from both patients and healthcare professionals, by the National Authorities of each member country of the PIDM. An ADR is a term used to describe the unintended side effects of drugs that pharmacovigilance experts are trying to detect.

When expressing an ADR in free text, the very same reaction can be explained in many different ways: "I have a headache", "my head hurts" and "I have a pain in my head" are all verbatim expressions of the same condition: Headache. It is of great importance to classify these expressions as equal in order to find possible correlations between drugs and side effects. In order to do this, normalization can be performed, mapping the different verbatim expressions to the same condition label. As of today this mapping is done manually by coding specialists who choose a fitting label based on the verbatim. The labels used for mapping are the terms found in The Medical Dictionary for Regulatory Activities (MedDRA) which is a terminology that contains several

10,000 of medical terms. This manual mapping can be a very time consuming task and requires the work of specialists. When leaving this task for human evaluation there is also the aspect of subjectivity which can result in similar verbatim being mapped to different labels by different coding specialists.

This work aims to develop an algorithm that can automate the process of mapping verbatim expressions of ADRs to MedDRA terms. The results of this project can be beneficial for the supervisor in multiple ways. For example in improving their mapping of verbatims in the side effect reports. At the time they only rely on direct matches to MedDRA and therefore might be missing valuable information. Both in pharmacovigilance and clinical trials experts are manually performing data entry. The proposed algorithm could therefore be a resource to improve these processes.

2. Background

The Medical Dictionary for Regulatory Activities (MedDRA) is a terminology that contains several 10,000 of medical terms presented in a hierarchical order containing five layers displayed in figure 1.[4] MedDRA is continuously updated with new medical concepts being added or existing concepts being modified. In this paper MedDRA version 22.1, released in September 2019, is used. [5] In this version the highest level layer of MedDRA 'System Organ Class' (SOC) contained 27 terms while the lowest level layer 'Lowest Level Term' (LLT) contained over 80,000 terms. The highest level layer (SOC) contains the most general terms and for each layer the terms get more specific. The 'Preferred term' (PT) is the term used to label the side effects in VigiBase and it consists of almost 24,000 terms. The most specific term (LLT) can contain for example synonyms or different spellings of the PT, as well as the PT itself.

Every PT is primarily assigned to one SOC but can also be secondarily assigned to several other SOCs. An example is the PT "Asthma" that is found under its primary SOC "Respiratory, thoracic and mediastinal disorders" (SOC) but also "Immune system disorders" (SOC) as secondary. Each LLT is however uniquely related to one PT.

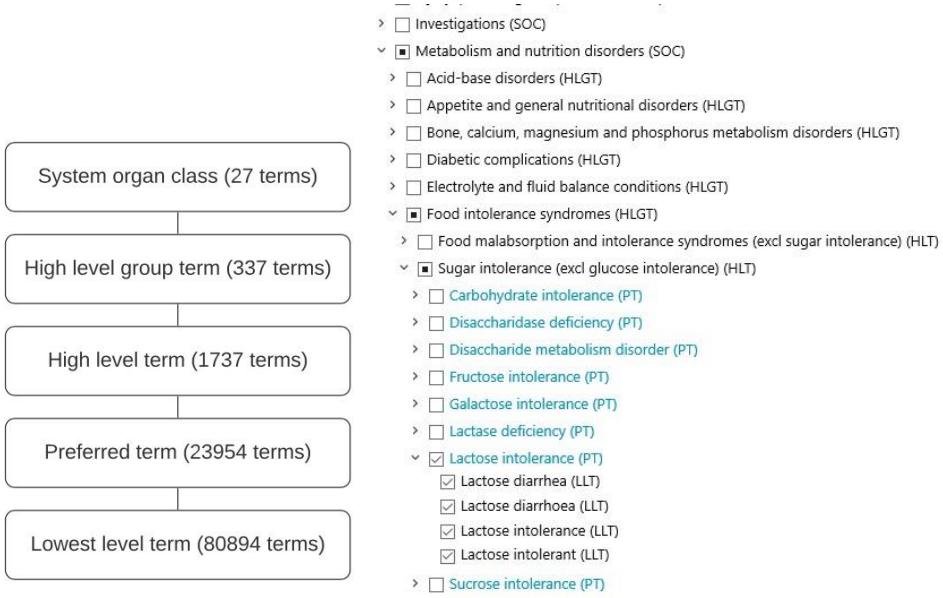

Figure 1: The five layers of the MedDRA hierarchy.Figure 2: An example showing the MedDRA hierarchy for the PT 'Lactose intolerance'.

Figure 2 shows an example of how 'Lactose Intolerance' is categorized in MedDRA. First we have 'Metabolism and nutrition disorders' (SOC) which is a very general term, afterwards comes the more specific 'Food intolerance syndromes' which is a 'High level group term' (HLGT). From the HLGT of food intolerance we specify even more unto 'Sugar intolerance (excl glucose intolerance)' which is a 'High level term' (HLT). Under this category we find the PT 'Lactose intolerance' that we were looking for. As can be seen the chosen PT has four corresponding LLTs. Among the LLTs we find the PT 'Lactose intolerance', 'Lactose intolerant' as well as synonyms including different spellings 'Lactose diarrhea/diarrhoea'.

3. Methodology

3.1 Natural Language Processing

Natural language processing (NLP) concerns the interaction between the human language and computers. In theory, translating human language to computers could be an easy task: words are just collections of characters and sentences are collections of words. However it can be more difficult in practice since the human language can be ambiguous as well as ever changing and evolving. Yoav Goldberg states in his book Neural Network Methods for Natural Language Processing that" People are great at producing language and understanding language, and are capable of expressing, perceiving, and interpreting very elaborate and nuanced meanings. At

the same time, while we humans are great users of language, we are also very poor at formally understanding and describing the rules that govern language." [6]

Classification is the technique of categorizing data to a set of discrete output values, referred to as classes or labels. The classification algorithm is created from patterns found in pre-existing mappings. By finding the features that create these patterns and determine how the data is mapped, any new observations can be mapped according to this information. The goal is to find enough features to be able to correctly map any new (unknown) data to the correct class. In an example of classifying vehicles to the labels "bus", "car" or "motorcycle" the features could for example be the number of wheels, the length of the vehicle and the presence of a steering wheel. An unknown vehicle with more than 4 wheels should probably be classified as "bus" while an unknown vehicle without a steering wheel should be classified as "motorcycle". These patterns could be found by looking at multiple buses, cars and motorcycles and finding how they distinguish from one another[10]

Classification is seen as a supervised learning technique in machine learning since it uses previously made classifications to make future predictions. The pre-existing mappings are referred to as the gold standard (GS) and they are seen as the benchmark. Within the NLP-field the data used is in text format, usually as words or sentences. When classifying data in this form the task is more specifically referred to as text classification. Some well-known examples are: sentiment analysis (text classified as having positive or negative sentiment) and language detection (predicting what language the text is written in)[7].

A. Multi-class classification

The simplest form of classification is called binary classification and is done with only two classes. Examples of binary classification are classifying data to True/False based on some criterion, for example classifying e-mails to being spam/not-spam or reviews of a product to being positive/negative.

When there are more than two classes it is considered a multi-class classification problem. As the number of classes increases the classification problem gets increasingly difficult to solve. To explain this increasing difficulty we can compare a binary classification problem (2 classes) with a multi-class problem of 100 classes. To exemplify the problem a" dummy classifier" could be used, that simply classifies everything to the same class. Statistically (not considering imbalanced classes) this would mean that in the binary classification we get 50% accuracy while in the case of 100 classes we get 1% accuracy. More generally we would get accuracy = nrOfClasses1, clearly showing the relation between a decreased accuracy with an increased number of classes. In practice the algorithms are usually better than this" dummy classifier", but as the number of classes increases any algorithm will have more outputs to consider, decreasing the possibility for a correct classification.

B. Imbalanced classes

Having imbalanced classes means that the number of observations from different classes, used to train the classifier, is disproportionate. This can lead to bias within the model as it is trained to classify more often towards the most represented classes which can give results that seem more promising than they really are. Let us say there is a binary classification problem where the goal is to detect spam e-mails and the observations are 95% non-spam and 5% spam. The"dummy classifier" that always classifies to nonspam would then give an accuracy of around 95%, which seems great, even if nothing has really been implemented[11].

C. Multi-label classification

Commonly within classification each observation is mapped to a single class. With multi-label classification however the observations are mapped to a set of classes, one or multiple ones.

D. Hierarchical classification

Usually all classes are equally differentiated from one another. If however the classes are part of a hierarchy they will be more or less related. This relation can be used with a hierarchical classifier that can start mapping data to a low-level and increase the level of detail.

3.2 Text representation

For any classification problem the input needs to be numerical since that is the only representation that a computer can comprehend. When working with NLP-problems we are using text as input and before doing any calculations we need some method to translate the text into numbers. There are multiple proposed solutions for text representation some of which are presented in this section.

A. Bag of words

One simple approach of representing text is Bag Of Words (BOW) which takes the words and its number of occurrences in a document into account. If two documents consists of the same words, they are seemingly similar and could therefore belong to the same class. By creating vectors that reflect on the term frequency, similarities between documents could be found by vector comparisons.

The data representation will be a vector were each position corresponds to a word that is present in some of the documents. Each document will then have their own vector were each number represents the occurrence of the word in the document. This way of representing each word with a vector of N positions with a "1" in the position representing that word and "0" for the other N-1 positions is called a one-hot encoding.

Document 1 = "Headache"

Document 2 = ”Drug exposure during pregnancy”
Document 3 = ”Drug exposure”
Using the three documents above as an example the corpus used would consist of the words: ”drug” ”during” ”exposure” ”headache” ”pregnancy”. Since the corpus consists of five words, the vector representation will be five dimensions. Comparing the vector representation below it is clear to see that document 2 has more in common with document 3 than document 1, as expected.
Document 1 = ”Headache” = [0 0 0 1 0]
Document 2 = ”Drug exposure during pregnancy” = [1 1 1 0 1]
Document 3 = ”Drug exposure” = [1 0 1 0 0]

B. Tf-idf

The Bag of words representation is based on term frequency but it doesn’t take into account the fact that words are more or less commonly used. Some words like ”the”, ”of”, ”a”, ”that” appear more often in the English language and these words might have a high frequency in multiple documents, even if these documents should not be seen as similar. ”Term frequency - inverse document frequency”, often shortened tf-idf, deals with this weakness by weighting the frequency of each term with the number of documents where they are present.

C. Word embeddings

There are multiple problems with the earlier mentioned representations (BOW and tfidf). One being the high dimensionality of the vectors, which will be growing with the number of terms in the corpus. Another problem is that similar words are not connected in any way; these representations lack awareness of word meaning. With BOW and tf-idf there will probably be similarities found between the vector representation of ”I feel pain in my head” and ”I feel pain in my arm” but not between ”she felt pain in her head” and ”he had a headache” since the last sentences have no common words.
In his book Speech and Language processing Jurafsky mentions terms such as word similarity and word relatedness [7]. These concepts can be used to understand the insufficiency of using term frequency for representation, which is simply based on the words and not the meaning behind them. Even if two words are not synonyms they can still be more or less similar or related to one another. Cat is not a synonym of dog, but cats and dogs are still similar words used in similar contexts. In the same manner coffee and cup are neither synonyms nor similar words, but they are still related and associated to one another.
Word embeddings are a collection of techniques used for creating word vectors and they often include the use of neural networks. This results in vector representations that are much denserthan the one-hot encodings mentioned in previous sections. Another advantage of using word embeddings is that the vectors capture semantic meaning of words from the contexts of where it appears. When training a word embedding model with sentences, not only the target word is considered but also its surrounding words. In 2013 Google introduced a word embedding model which later became known as word2vec [8] that became very popular for creating word vectors. It makes use of two architectures called CBOW and Skip-gram and produces word vectors from unsupervised training on a large text corpus.

3.3 Machine learning

Machine learning is a field within computer science. The objective is for the computer to” learn” how to solve a problem (that it is not explicitly programmed to solve) based on data. There are different kinds of machine learning algorithms which can be divided into separate categories. The three most common ones are supervised learning, unsupervised learning and reinforcement learning.
Supervised learning can be used if we have access to a labeled data set of observations. The model can learn from this set and find patterns that will help make future predictions. When there is no labeled dataset to begin with, unsupervised learning algorithms can be used. These models tries to group data together based on underlying patterns. Lastly reinforcement learning is based on interaction with the environment. The system learns by rewards, were better choices are rewarded higher and thereby effecting future choices.

3.4 Deep learning

Deep learning is a sub-field within machine learning that is based on artificial neural networks (ANNs). ANNs are a set of algorithms with a structure inspired by the signal transmission of neurons in the brain. An ANN is built in multiple layers: starting with the input layer, ending with the output layer and then a number of hidden layers in between. ANNs operates on numerical data and the input must be of fixed size. When working with data that is not numerical by default, for example text, it needs to be translated into a numerical data representation. When using deep learning algorithms the features are extracted from the data without human intervention, as opposed to traditional machine learning algorithms. This however comes at a cost of needing a relatively high amount of training data for the algorithm to be successful.

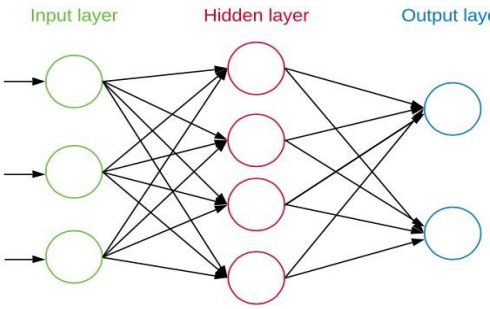

Figure 3: The figure shows the structure of a simple ANN with a single hidden layer.

Recurrent neural networks (RNNs) is a field within neural networks with algorithms that, opposed to the simpler neural networks, take sequential structures into account (through time or space depending on the application). This is accomplished by internal feedback loops in the network that creates what can be referred to as ”memory”.

In a paper from 2017 [10] researchers from Google presented the concept of Transformers, built in an encoder-decoder architecture. The conventional encoder-decoder model has a sequence of connected RNNs where each RNN inputs a token and outputs a vector that is based on the token as well as all the previous tokens. One disadvantage of this model is that the input has to be fed to the system sequentially, as each step is dependent on previous calculations. The Transformer introduced an alternative to the RNN architecture, which uses something called attention instead of recurrence. As opposed to the RNN architecture with the sequential dependency of the input, the Transformer reads the entire word sequence at once and can learn its context both from the previous as well as the following words. The model is thereby considered to be bidirectional.

BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a pre-trained language model that was released by Google in 2018. As the name might reveal, the BERT architecture consists of multiple layers of transformer encoders. The model has received state-of-the-art results on multiple NLP tasks [9]. BERT is trained on a huge data set consisting of the English Wikipedia (2,500M words) and the BooksCorpus (800M words). The training is unsupervised and two different tasks are performed, namely Mask language model (MLM) and Next sentence prediction (NSP), which creates the word representation. For MLM the model looks at an entire sentence or piece of text trying to predict the word that has been masked out. This task is seen as a key technical innovation as it uses the bidirectional training of Transformers for a language modelling task. For the NSP task, the model received pairs of sentences and had to predict if the second sentence followed the first in the original document.

The pre-trained BERT model has a general knowledge of the English language but the model needs to be fine-tuned to perform a specific task. For a classification task this means training the model using training data to detect how the input relates to the classes.

As input BERT takes a sequence of tokens. BERT has a corpus of tokens that can be numbers, words or segments of words, which is used to represent the input. One benefit of the token representation is that any word can be represented. If a word is not found within the corpus it can be broken down into multiple tokens and possibly keep some of the original meaning. As an example BERT has no single token to represent the word ”chills” but it can instead be represented with the tokens ”chill” and ”##s”, where the ”##” represents that the token is part of the same word as the previous token. If the word” headache” was not found in the token corpus it would be represented by the tokens” head” and” ##ache” which in this case still contains most of the meaning. The worst case scenario would be for a word to be broken down into each character that it consists of: not keeping a lot of meaning but still being able to represent the word. The number of tokens is fixed length for a given model and corresponds to the size of the input layer. If the number of tokens for a given model is 64 it means that any input that can be represented by less tokens will be padded using a [PAD] token and if an input needs more than 64 tokens it will be cut off.

When using the fine-tuned BERT model for classification it produces an output layer of logits, with a layer size corresponding to the number of classes. Logits are integer values ranging from $-\infty$ to ∞ and represents the unnormalized predictions of the model. The logits can be turned into a distribution of probabilities using the SoftMax function. Each class will then be represented by a value showing the models probability for each possible class being the correct class of a given input. When using the SoftMax function on the logits layer to create a vector of probabilities, taking the sum of that vector will always add up to 1. The SoftMax function is defined as:

Where S is the resulting SoftMax vector, e is the standard exponential function and y is the vector of logits ranging from position i = 1,2,...n.

3.5 Data division

In machine learning tasks it is a common practice to split the obtained data set into separate disjoint subsets for training, validation and testing. The training set is used to fit the model. It is from this set that the model learns patterns. The validation set is used to repeatedly evaluate the model. Since the validation set is disjoint from the training set it will provide new data for the model. From the results of the evaluation of the validation set, parameter tuning can be performed. In conclusion the model indirectly learns from the validation set. Lastly the test set is used to evaluate the final model. No further changes should be made to the model after evaluating against the test set as these results are seen as the actual performance of the model.

4. Data

Data was retrieved from a frozen version of VigiBase, containing all reports up to 5th of January 2020. From this source *9,869,169* rows of annotated data could be retrieved, each containing one verbatim and one label. The verbatims are the inputs to our system, the observations that we want to classify. They are freely entered text containing a single up to multiple tens of words, expressing ADRs in multiple different languages. The labels are the preferred terms (PTs) from the MedDRA hierarchy that the verbatims have been mapped to. This mapping is done by an expert while making the ICSR. MedDRA is, like VigiBase, ever changing. In this project MedDRA version 22.1 was used [5]. In this version there are 23,954 PTs, however in our labeled data set only 16,408 (∼68,5%) are representedAn initial approach was exploring the data to get a feeling of what kind of difficulties there were and what methodologies could be fitting to solve them. An early finding was that even as the verbatims were pulled from free text fields, they were not all unstructured. A large number of verbatims were already in the form of LLTs, the lowest level term of the MedDRA hierarchy.The verbatims are expressed in free text and can contain all sorts of characters. In the data there exist alphabetical characters, special characters and numbers. They can contain abbreviations and spelling mistakes to list a few. Among the English verbatims a common use of non-alphabetic characters are numbers used to report on medical measurements. Examples of verbatims including such measurements are ”mxd raised 1.7x10ˆ9/l”, ”lipase (over 4000u/l)” and ”high white blood cell count 80”.The verbatims in the given data set are of varying length, consisting of between 1 to 53 words. However, around 70% of the verbatims only contain up to 3 words. For the purpose of word counting a word was defined to be a number of characters separated from other characters with spacing. Each row of data is only mapped to one class, although the verbatim can contain multiple reported side effects. This means that even though there will only be a single correct class for each row of data (that is our gold standard), there might be multiple fitting classes for the verbatim. This will complicate our classification process and is something to keep in mind when evaluating the algorithm.As presented in the PT-level in MedDRA contains 23,954 terms. Having this enormous selection of labels means dealing with an extreme multi-class problem. When examining the annotated data the imbalances of the classes is a fact. Looking at the training data only 13,978 classes are represented, meaning that we already lost ∼42% of possible classes. In table 1 the imbalances of the data becomes clear. Even if the training data set contains 13,978 classes, the table shows that only the 100 most common classes are used to label 54,9% of the data set. This implies that the 45,1% left is split between the other 13,878 classes in different measures.

Table 1: This table shows how much of the training set that is covered for a number ofPTs most commonly used for labeling this set.

Number of PTs	Data covered
100	54,9 %
500	80,7 %
1,000	89,1 %
2,000	94,9 %
3,000	97,1 %
4,000	98,2 %
5,000	98,8 %
7,000	99,5 %
10,000	99,8 %

A. Language Filtering

Since the data consists of verbatims from ICSRs retrieved from countries all over the world, multiple languages will be present in the data. This work will be limited to working with English verbatims and therefore it is important to find a method that can successfully separate the English from the non-English verbatims. A number of methods where tested to find the most effective one for this task. In this section the different methods are presented. The methods were all evaluated on the same set of 2000 data rows that were randomly sampled from the whole data set. The 2000 verbatim were manually labeled as ”English” or ”non-English”, resulting in 478 non-English and 1522 English verbatims.

B. Sort by country

An early and easy approach was to examine if we could simply choose to include data from countries where a majority was written in English. The countries were chosen based on a manual overview of the data. The countries chosen were: United Arab Emirates, Australia, Canada, Egypt, United Kingdom of Great Britain and Northern Ireland, Greece, India and Korea. However, this method led to a giant data loss since many countries had to be excluded because of the presence of non-English verbatims, even though there was a lot of useful english data also present in these data sets.

C. Regular expression

When examining the differences in the English vs Non-English data it was clear to see a common difference in the characters present. There was data containing only nonalphabetical characters (Chinese/Japanese) and data containing vowels like" a, ` e´ and ¨ı" (French/Italian). By using regular expression these rows could be found and discarded.

D. Python libraries

The python library langid was used to remove some non-English rows. The function langid.classify(verbatim) was invoked for every verbatim. Every call made with a verbatim that resulted in "en" (classified as English by langid) was kept and all other verbatim were removed.

E. Dictionary

A dictionary was created using a combination of all words in the lexical database WordNet [11] and all the words present in any Lowest Level Term in MedDRA (any numbers or special characters where not added). When evaluating this method each verbatim was split up into separate words and each word compared against the dictionary. The verbatim then received a score of $score = {}^{englishWords}{}_{words}$ where *words* refer to the number of words in the verbatim and *englishWords* the number of words in the verbatim found in the dictionary. Finally, since the goal of this language sorting was to make sure that all data we operate the project on is in actually in English, we based the threshold of the score upon the precision. The precision was set to be .99 allowing for a .01 error rate of non-English verbatim.From evaluation of the results the final choice for retrieving only the English data was a combination of the *Dictionary* and *Regular expression* methods mentioned above. Using this method left 6,986,110 rows of data (∼70% of the original set).

F. Modules

The modules are different approaches taken to solve the problem of this work. They are based on simple string matching algorithms as well as the more advanced technology of the BERT model. The reason for creating these modules was to compare the results of different algorithms as well as exploring if a combination of different algorithms would be more beneficial for solving the problem.

i. String matching - LLT

From explorations of the training data it was discovered that there were many verbatims that were already written as MedDRA terms. To further explore this finding, a string matching algorithm for classification was constructed. The algorithm was designed to compare each verbatim to a dictionary consisting of all LLTs. All the LLTs were lowercased to match the lowercased verbatims. If a match was found, that verbatim would be classified to the LLTs corresponding PT.

ii. String matching - training

The second algorithm used the same approach of string match comparisons, but comparing the verbatim we want to classify to the verbatims in the training data. We wanted to make use of previous data by classifying accordingly. If any match was found, the verbatim would be labeled as it was labeled in the training data. An initial problem with this approach was that the very same verbatim expression can be labeled differently in the training data, resulting in multiple labeling options. To solve this problem all the verbatims in the training data was compiled into a dictionary of distinct verbatim expressions. Each distinct verbatim worked as a dictionary key connected to a PT label value. When creating the dictionary, each distinct verbatim would get the PT that it was most commonly labeled as in the training data.

iii. BERT

We used a pre-trained BERT base model and fine-tuned it for our classification task using the 4,890,274 rows present in the training data set and trained for 4 epochs. The input layer was set to 32 and the output layer to 5,000, meaning we input 32 tokens and have 5,000 possible classes as output. As can be seen in table 1 considering the whole data set the 5,000 most commonly used labels covers 98.8% of the data labeling.When using the fine-tuned BERT model for classification it produces an output layer of logits, with a layer size corresponding to to the number of classes, in our case 5,000. We take the SoftMax of the logits and classify the verbatim to the class with the corresponding logit of highest SoftMax value.

Thresholds

Because the verbatims are constructed in free text fields some might be very difficult to classify. They could for example include measurements, abbreviations, multiple symptoms or other ambiguity. In order to avoid misclassification, one option would be to not classify the most difficult verbatims. The values of the logit layer

reflects on the confidence of the BERT model making good predictions. By taking the values of the logits into account in the classification process we could decide how confident we need the model to be.
Since the value of the logits represent the confidence of the corresponding class being correct, we decided to use this value as a threshold. This is referred to as the valthreshold. As discovered in the explorations there are cases where multiple side effects are reported in the same verbatim. This could lead to multiple logits getting high values. To increase the certainty of the prediction we make, we chose to also include another threshold based on the difference between the highest and second highest logit. The smaller the difference between the two highest logits, the less certainty that the highest value results in a correct prediction, as we have high confidence in multiple classes. The second threshold is referred to as the diff-threshold.
In practice each verbatim is classified with the PT corresponding to the highest logit value, if the logit value exceeded the valthreshold and the difference between the highest and second highest logit values exceeded the diff-threshold. For any verbatim for which the logit values do not satisfy the thresholds, no classification is made. But even if the confidence of BERT's prediction is not seen as good enough to classify a verbatim, there might still be good suggestions among the top predictions. With any verbatim that is left unclassified the top 5 highest ranked PTs (based on the highest logits from the BERT output layer) will therefore be provided. If these verbatims are left for manual mapping it means that the 5 suggestions could be a resource in the process.
Three different modules of BERT were created with the thresholds based on the maximum f_1-score, the maximum $f_{0.5}$-score and the maximum $f_{0.2}$-score for both the highest value logit (valthreshold) and the difference between the highest and second highest logit (diff-threshold).

G. Evaluation

The modules were evaluated in different combinations to find a pipeline of modules that gave us the best results on the validation set. The pipelines were evaluated by accuracy (number of correct predictions) and error rate (number of incorrect predictions). If this pipeline were to be used in clinical trials it would be very important to not be making incorrect mappings. Because of this we want to keep the error rate as low as possible.
The evaluation of the pipelines was based on comparisons between the predictions and the gold standard (GS) as well as an error analysis performed on the different modules of the final pipeline. The basis for the error analysis was produced by a panel of terminology specialists at UMC. They were asked to review 200 randomly selected verbatims from each module, where the predicted PT was different from the GS. The terminology specialists looked at each verbatim and chose a PT that they would code that verbatim to. Each verbatim was then given a label that shows how the terminology specialists' PT relates to the (by our system) predicted PT and the gold standard PT.
In table 2 the possible labels and their corresponding description are shown. TS refers to the PT chosen by the terminology specialists, GS refers to the PT that is our gold standard and P refers to the (by our system) predicted PT. The label"-1" was given when the specialists felt there was not enough information in the verbatim to give it a PT label. "0" was given when the specialists chose a PT that was not predicted by our system, neither the gold standard. "1" is the case when the specialist chose the same PT as our system predicted, and "2" when they chose the same as the gold standard. The last label "3" was chosen when the verbatim contained information linking to multiple PTs, were the specialists would split up the verbatim and code the parts separately.
As mentioned in the output of the manual mapping module is five suggested PTs. In this evaluation "0" was given when none of the five suggestions (or the GS) was the same as the specialists' choice and "1" when one of the five suggestions matched their chosen PT.
Table 2: The labels used for evaluating the incorrect samples from the different modules

Label	Description
-1	The verbatim has no fitting label
0	(TS != P) AND (TS != GS)
1	TS = P
2	TS = GS
3	The verbatim should be coded to multiple labels

5. Results

5.1 Language sorting

The results of methods presented for sorting out English data is shown in the table 3. The different methods were evaluated on a set of 2,000 randomly sampled rows which were manually labeled as English or non-English.From table 3 we find that the method *Countries* had a high precision of almost .99, but a low recall of 71. It shows that when data was solely selected from a few countries, the selected data was mainly in English. However a lot of English data (from other countries) were filtered out.The *RE* approach had a prefect recall of 1

meaning that all the English samples were classified as English. The low precision of around .78 however shows that this method did not filter out non-English data strictly enough as there were still much left in the data set.
Python LangID had relatively good results in both precision (.92) and recall (.89) but was out-performed by the *Dictionary* method that got a precision of .99 and a recall of.96.When combining the *Dictionary* method with the *RE*, having a perfect recall, the precision was slightly improved without any negative effect on the recall. This led to the best precision and fscore of all the methods which led to the final choice of the *Dictionary + RE* as the method for language sorting.

Table 3: The different methods for distinguishing English verbatims, evaluated with a sample set of 2,000 manually labeled rows of data.

Method	Precision	Recall	F_1-score
Countries	0.9899	*0.7063*	*0.8244*
RE	*0.7841*	1	0.8790
Python LangID	0.9192	0.8890	0.9038
Dictionary	0.9898	0.9560	0.9726
Dictionary + RE	0.9905	0.9560	0.9729

5.2 Thresholds

Table 4 shows the three different thresholds used for the BERT module. Each one of these three thresholds corresponds to one ”diff”-threshold and one ”val”-threshold. The values used for” diff” (difference in highest and second highest logit) and ”val” (the value of the highest logit) were chosen because they were maximizing three different f-scores (F_1-score, $F_{0.5}$-score and $F_{0.2}$-score).

Table 4: The different thresholds used to improve the predictions made by BERT

Threshold	Diff	Val
F_1	0.4	9.2
F0.5	2.0	10.8
F0.2	4.1	12.6

When evaluating the validation set on the BERT module, the distribution of the correctly classified verbatims are displayed to the left in figure 4. The y-axis shows the value of the highest value logit (referred to as ”val”). The x-axis shows the difference between the highest and second highest value logits (referred to as ”diff”). To the right in figure 4 the distribution of the incorrectly classified data is displayed. It may look as if the graph showing the incorrect predictions has more data because of the intensity of the heat map. However this is a result of the graphs being generated separately, thereby the intensity is not comparable.
Comparing the distributions in the graphs, the incorrectly classified verbatims are much more centered towards the lower values of both ”diff” and ”val”, while the correctly classified verbatims are more centered around higher values. The three different boxes present in both graphs show how the three different thresholds introduced in table 4 affect the number of occurrences of correct and incorrect predictions in the validation set. Everything inside the box will be left unclassified for that specific threshold, while everything outside the box is classified by BERT. As can be seen the higher the threshold boundary, the fewer incorrect predictions are made. However, this also means that more of the correct predictions will be left unclassified. The overlapping distribution of the two graphs shows that no threshold will completely eliminate the errors. Choosing a threshold will really be a trade-off of getting the best possible accuracy without overstepping the accepted error rate.

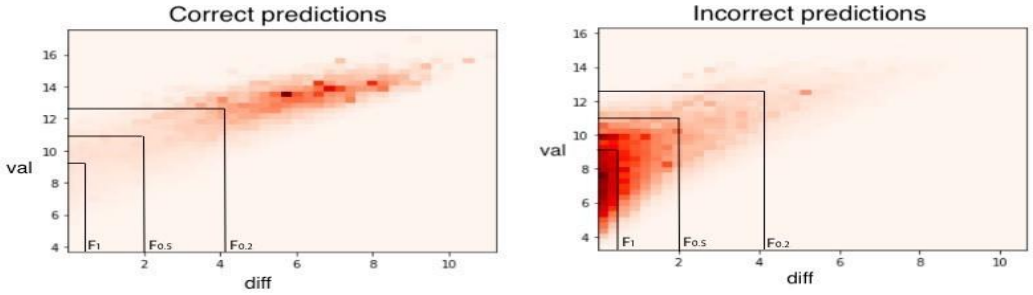

Figure 4: Two heat map graphs displaying the data distribution from the by BERT correctly and incorrectly predicted verbatims. The graphs also show the three different thresholds from table 4. The graphs were generated separately and the intensity can therefore not be compared.

5.3 Pipeline

A number of possible pipelines were evaluated based on different combinations of modules presentedwith different thresholds presented in table 4. The pipelines are shown in table 5 and the evaluation is done on the validation set.In *pipeline 1* the BERT model is classifying the whole validation set. The accuracy is 0.8376 which can be interpreted as a good result considering the difficult examples that exists in the data set. However the drawback of this pipeline is the high error rate of 0.1624, which would never be accepted for the potential use case of the system. To let BERT classify every verbatim seemingly was a too simple approach. When restricting the classification with different thresholds in *pipeline 2-4* both the accuracy and the error rate drops as the threshold boundary is increased.

The results of *pipeline 5* shows that more than half of the verbatims in the validation set are actually in the form of LLTs. Even as the error rate for this pipeline setup is relatively low (around 1%) the presence of these errors is still surprising. They occur when a verbatim, that is also a LLT, is mapped to another PT than the one that the LLT is corresponding to.When apart from LLTs also basing the predictions on training data (meaning classifying a verbatim in the validation set as it was classified in the training set) the error rate increases from 1,1 % to 3,58% (comparing *pipeline 5 and 6*). This shows that the exact same verbatim can be classified differently, which strengthen the hypothesis that the mapping is sometimes based on additional information. Because of the high error rate brought by basing future predictions on exact matches in the training data (as can be seen in *pipeline 6, 8 and 9*) this module was not kept in the final pipeline.
Pipeline 7 is a combination of the modules separately run in *pipeline 4 and 5*. When comparing these pipelines it is found that the combination really improves the accuracy without increasing the error rate notably. The improved accuracy of adding the LLTmatching before predicting with BERT can possibly be a result of BERT being limited to a number of classes (in this case 5,000). The LLT-matching can predict any of the almost 24,000 classes as long as the verbatims are in the form of LLTs, which a notable section seemingly.
Pipeline 10 can also be compared to *pipeline 7*. The pipelines consist of the same modules but using different thresholds for the BERT prediction. With the relaxed threshold in *pipeline 10* the accuracy approximately increases from 71% to 79%. However *pipeline 7* was considered a better option as the error rate more than doubled with the relaxed thresholds.

Table 5: Evaluation of possible pipelines on the validation set

id	Module 1	Module 2	Module 3	Accuracy	Error rate	Unclassified
1	BERT	-	-	0.8376	*0.1624*	0
2	BERT F_1	-	-	0.8054	0.0714	0.1232
3	BERT $F_0.5$	-	-	0.7520	0.0334	0.2146
4	BERT $F_0.2$	-	-	0.6258	0.0111	0.3631
5	LLT	-	-	*0.5549*	0.0110	*0.4341*
6	LLT	Training	-	0.7046	0.0358	0.2596
7	LLT	BERT $F_0.2$	-	0.7106	0.0169	0.2725
8	LLT	BERT $F_0.2$	Training	0.7649	0.0387	0.1964
9	LLT	Training	BERT $F_0.2$	0.7612	0.0389	0.1999
10	LLT	BERT $F_0.5$	-	0.7923	0.0365	0.1712

5.4 Final model

For the final model, pipeline number 7 from table 5 was chosen because of its low error rate and relatively high accuracy. In figure 5 the details of how the selected pipeline works are shown. The first two modules *LLT* and *BERT* are the automatic part of the pipeline that perform classification. The resulting correct and incorrect classifications are displayed in the figure. The data that could not get automatically classified is left to the last module *Manual mapping* where the 5 top suggestions from BERT are given. The figure shows if the correct label is found within the top 5 suggestions or not.
When evaluating the pipeline on the validation set the accuracy is measured to 71.06% (LLT-matching: 55.49% and BERT: 15.57%) compared to the 80,21% (LLT-matching: 66.50% and BERT: 13.71%) when evaluating on the test set. The increase in accuracy can evidently be explained by a bigger part of the test set consisting of LLT terms, compared to the validation set. The sets were split over time and the increase in LLTs among the later received verbatims could be explained by how newer reporting systems choose to input this information.
The results of the manual mapping module shows that the suggestions generated by BERT are rather accurate. For the validation set about 77% of the verbatims left for manual mapping has the correct label found in the top 5 suggestions, for the test set the corresponding results were about 82%. This shows that the suggestions could actually be a useful resource for someone who were to map these verbatims manually.

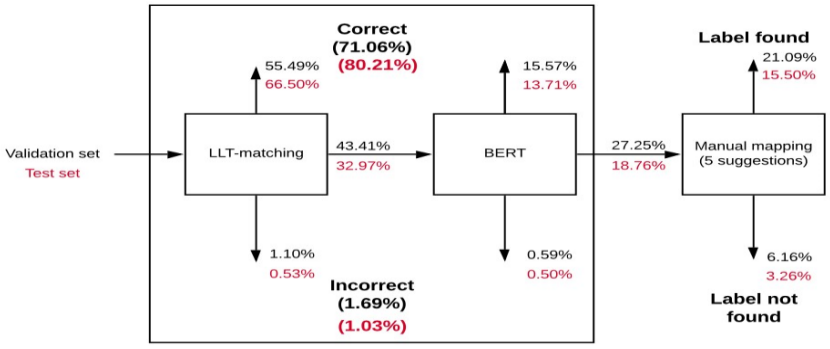

Figure 5: Evaluation of the final pipeline.

The results on the validation set are shown in black while the results of the test set are shown in red. The percentages are given as fractions of each entire data set.

5.5 Classification examples

Table 6, 7 and 8 shows examples of data from the test set that is incorrectly predicted in the different modules of the pipeline. The examples were selected to show as many different scenarios of wrong predictions and classifications as possible.

Table 6 displays verbatims in the form of LLTs that was incorrectly classified. The verbatim *pregnant* is classified as *Pregnancy* but was labeled in the validation set as *Exposure during pregnancy*. Seemingly the label contains more information than the verbatim, making the classification, which is solely based on the verbatim, very difficult. Another example from the same table is the verbatim *feels bad* that is labeled *Malaise* but classified as *Feeling abnormal*.

In table 7 the classification *Lipase increased* made by BERT for a verbatim labeled as *Hyperlipasaemia* is found. The classification is incorrect as the correct label was not captured, however hyperlipasaemia is a diagnosis given for someone with increased lipase. A similar case, found in the same table, is a verbatim that is classified as *Pyrexia* (which is the diagnostic term for having a fever) while the label is set to *Body temperature increased*.

Presented in table 8 are examples of verbatims that were not classified by BERT and where the label was not found in the top 5 suggestions. The verbatim *swollen tongue, dyspnoea, dysphagia, drooling, cough* is labeled as *Drooling*. In the top 5 suggestions we find labels such as *Dyspnoea*, *Cough* and *Swollen tongue*, which are all symptoms expressed in the verbatim.

Table 6: Examples of data incorrectly classified by LLT-matching

Verbatim	Label (PT)	Classification (PT)
Pregnant	Exposure during pregnancy	Pregnancy
application site reaction	Skin reaction	Application site reaction
painful rash	Pain of skin	Rash
feels bad	Malaise	Feeling abnormal
bunion operation	Foot deformity	Bunion operation

Table 7: Examples of data incorrectly classified by BERT

Verbatim	Label (PT)	Classification (PT)
lipase (over 4000u/l)	Hyperlipasaemia	Lipase increased
high temperature all over the body	Body temperature increased	Pyrexia
Imbalance	Walking disability	Balance disorder
increased anger	Mood swings	Anger
does not have af	Off label use	Atrial fibrillation

Table 8: Examples of data that was unclassified by BERT (because the logit values did not pass the thresholds) where the correct label was not found in the top 5 suggestions.

Verbatim	Label (PT)	Top 5 suggestions (PT)
multiple tumors in mediastinum	Mediastinum neoplasm	['Colon cancer', 'Malignant neoplasm progression', 'Neoplasm malignant', 'Second primary malignancy', 'Neoplasm']
tablet breakage	Prescribed underdose	['Product physical issue', 'Product quality issue', 'Wrong technique in product usage process', 'Product container issue', 'Product complaint']
raised ketone	Acetonaemia	['Blood ketone body present', 'Urine ketone body present', 'Ketoacidosis', 'Dehydration', 'Bone disorder']
fracture (traumatic)	Femoral neck fracture	['Fracture', 'Upper limb fracture', 'Fall', 'Stress fracture', 'Multiple fractures']
swollen tongue, dyspnoea, dysphagia, drooling, cough.	Drooling	['Dyspnoea', 'Cough', 'Swollen tongue', 'Anaphylactic reaction', 'Dysphagia']

5.6 Error analysis

In this section the results of the error analysisare shown in table 9. For this evaluation 200 random samples of incorrectly predicted verbatims were chosen from each module (LLT-, BERT- and the manual mapping module) and classified by terminology specialists at UMC.

For the LLT module the specialists classified all the 200 verbatims with the same PT as our system. They also noted that in 105 of the 200 cases (just over 50%) the verbatim was the exact same as the PT chosen by them and our LLT module. These results are not surprising as the LLT module is based on direct string matches and should therefore hypothetically have a high accuracy. As mentioned previously, looking at table 6, the correct labels sometimes seems to be based on more information than solely the verbatim. In this error analysis however, the terminology specialists, similarly to our pipeline, only consider and have access to the verbatim.

For the BERT module in 149 cases (covering 74,5% of the samples) the specialists chose the same PT as BERT had predicted, compared to the 15 cases (7,5%) where they chose the gold standard PT. There were also a few cases were the verbatim could not be classified either because the specialists wanted to split it up as it should be mapped to multiple PTs (label "3") or because it could not be mapped to any PT (label "-1"). 7% of the set was labeled as"-1" and two examples of that are the verbatims:" to *be high"* and" does *not have af"*. Another 4.5% needed to be split up, one example is the verbatim: *"hyperglycemia/pain in hands/swelling on feet"*.

When it comes to the last module, the manual mapping module, the chosen labels are more distributed over the different options then for the other modules. In 27.5% of cases the specialist chose the same PT as one of the five suggestions from our system while in 26.7% they chose the same as the gold standard. Another 24.5% of the verbatims needed to be split up, 8% of verbatims could not be labeled and in 13.5 % of cases none of the five suggestions nor the gold standard matched the PT chosen by the specialists.

Table 9: The results of the error analysis are shown as fractions for each module and the number of occurrences of the labels are shown in parentheses. The description of the labels can be found in table 2.

Label	LLT module	BERT module	Manual mapping module
-1	0 (0)	0.070 (14)	0.080 (16)
0	0 (0)	0.065 (13)	0.135 (27)
1	1 (200)	0.745 (149)	0.275 (55)
2	0 (0)	0.075 (15)	0.265 (53)
3	0 (0)	0.045 (9)	0.245 (49)

6. Conclusion

The final pipeline design consists of three different modules namely the LLT-matching module, the BERT module and lastly the Manual mapping module. BERT is by far the most important and complex module as it

can make automatic predictions for all kinds of English verbatims. The shortcomings of the other modules are that the LLT-matching can only make predictions for verbatims structured as LLTs and the Manual mapping is not automatic as it involves human judgement. However through evaluation of the pipelines in table 5 it shows that all the above mentioned modules serves a purpose in the final pipeline, leading up to that choice.

The overlapping distribution of the correctly and incorrectly predicted verbatims, in the graphs displaying information from the output layer of BERT, shows that no threshold will completely eliminate the errors. Choosing a threshold is really a trade-off of getting the best possible accuracy without overstepping the accepted error rate. The threshold could be made more or less strict depending on the task of the model and its accepted error rate.

When comparing the correct labels with either the classification in tables 6 and 7 or with the corresponding suggestions in table 8 it is clear to see that one major problem for our model is the great number of classes and how similar they can be. We see that even if the classification or given suggestions are incorrect (as they differ from the actual label) the classes are in several cases similar in meaning. Another problem seems to be the limited information gained from solely using the verbatim as input, as some labels seems to be based on more information then what is given in the verbatim.

From the error analysis, it was discovered that, assuming that the verbatim is the only source of information, there are cases where the gold standard might not be appropriate. This could be a result of many things. It has been mentioned before that the gold standard might be based on additional information and as the mapping is done manually it would not be surprising for some human errors to occur as well. The specialists that performed the evaluation found cases of verbatims that, according to them, could not be mapped or that should be split up into multiple verbatims before being mapped. They also found that for many of the verbatims the model's classifications, which were evaluated as errors, were actually matching their own proposed labels. When looking at errors from the BERT module, the specialists had chosen the same label as BERT classified in 149 of the 200 samples (74,5%).

References

1. World Health Organization (2002), The Importance of Pharmacovigilance, Available at: https://apps.who.int/iris/bitstream/handle/10665/42493/a75646.pdf
2. World Health Organization, The WHO Programme for International Drug Monitoring, Available at: http://www.who.int/medicines/areas/quality safety/safety efficacy/National PV Centres Map/en
3. VigiBase, The unique global resource at the heart of the drive for safer use of medicines, Available at: https://www.who-umc.org/vigibase/vigibase
4. Mozzicato P (2009), MedDRA An Overview of the Medical Dictionary for Regulatory Activities, Available at: https://www.researchgate.net/publication/233524508 MedDRA An Overview of the Medical Dictionary for Regulatory Activities
5. MedDRA (2019), Introductory Guide MedDRA Version 22.1, Available at: https://admin.new.meddra.org/sites/default/files/guidance/file/000354 intguide 22.1.pdf
6. Goldberg Y (2017), Neural Network Methods for Natural Language Processing
7. Jurafsky D. and Martin J. H. (2019) Speech and Language Processing, Available at: https://web.stanford.edu/jurafsky/slp3/
8. Mikolov T, Chen K, Corrado G and Dean J (2013), Efficient Estimation of Word Representations in Vector Space, Available at: https://arxiv.org/pdf/1301.3781.pdf
9. Ramesh, (2020) "A Survey on NLP based Text Summarization for Summarizing Product Reviews," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 352-356, doi: 10.1109/ICIRCA48905.2020.9183355. (IEEE) (Scopus)
10. Dr. G. Ramesh (2020). A Survey on Hybrid Machine Translation, 2nd International Conference on Design and Manufacturing Aspects for Sustainable Energy (ICMED 2020), , Volume 184, August, 2020.
11. Dr. Gajula Ramesh (2020). Detection of Plant Diseases by analyzing the Texture of Leaf using ANN Classifier. International Journal of Advanced Science and Technology, 29(8s), 1656 – 1664.
12. Dr. G. Ramesh (2020). Data Storage in Cloud Using Key-Policy Attribute-Based Temporary Keyword Search Scheme (KP-ABTKS). Lecture Notes in Networks and Systems Volume 98 pp. 630–636, 2020.